A COMPARISON OF BIO-INSPIRED AND CONVENTIONAL ROUTING ALGORITHMS FOR FIXED-WING UAV FANETS

A THESIS SUBMITTED TO
THE GRADUATE SCHOOL OF INFORMATICS OF
THE MIDDLE EAST TECHNICAL UNIVERSITY

BY

BURAK ŞEKER

IN PARTIAL FULFILLMENT OF THE REQUIREMENTS FOR THE DEGREE OF
MASTER OF SCIENCE
IN
THE DEPARTMENT OF INFORMATION SYSTEMS

SEPTEMBER 2021

Approval of the thesis:

A COMPARISON OF BIO-INSPIRED AND CONVENTIONAL ROUTING ALGORITHMS FOR FIXED-WING UAV FANETS

Submitted by **BURAK ŞEKER** in partial fulfillment of the requirements for the degree of **Master of Science in Information Systems Department, Middle East Technical University** by,

Prof. Dr. Deniz Zeyrek Bozşahin
Dean, **Graduate School of Informatics** _____

Prof. Dr. Sevgi Özkan Yıldırım
Head of Department, **Information Systems** _____

Assoc. Prof. Dr. Altan Koçyiğit
Supervisor, **Information Systems, METU** _____

Assoc. Prof. Dr. Enver Ever
Co-supervisor, **Computer Engineering, METU-NCC** _____

Examining Committee Members:

Assoc. Prof. Dr. Pekin Erhan Eren
Information Systems, METU _____

Assoc. Prof. Dr. Altan Koçyiğit
Information Systems, METU _____

Assoc. Prof. Dr. Eser Gemikonaklı
Computer Engineering, University of Kyrenia _____

Date: 10.09.2021

I hereby declare that all information in this document has been obtained and presented in accordance with academic rules and ethical conduct. I also declare that, as required by these rules and conduct, I have fully cited and referenced all material and results that are not original to this work.

Name, Surname: Burak Şeker

Signature :

ABSTRACT

A COMPARISON OF BIO-INSPIRED AND CONVENTIONAL ROUTING ALGORITHMS FOR FIXED-WING UAV FANETS

Şeker, Burak

M.S., Department of Information Systems

Supervisor: Assoc. Prof. Dr. Altan Koçyiğit

Co-Supervisor: Assoc. Prof. Dr. Enver Ever

September 2021, 139 pages

Fixed-Wing Flying Ad Hoc Network (FW-FANET) indicates a communication network in which the nodes are fixed-wing UAVs, or drones, which are distributed within a 3D environment. In Fixed-Wing Flying Ad Hoc Networks (FW-FANETs), the network topology is very dynamic due to the mobility of nodes within a 3D environment. Hence, the selection of the routing algorithm is one of the main challenges. Among other alternatives, as a Swarm Intelligence-based bio-inspired routing algorithm, AntHocNet offers a great potential to deal with the issues of FW-FANETs. In this thesis, a bio-inspired routing algorithm, i.e., AntHocNet, and traditional routing algorithms, i.e., the AODV and OLSR, are compared in FW-FANET applications using the NS3 network simulation environment. This comparison examines how the increasing topology change rate and the increasing network load affect the performance of compared routing protocols. The analysis of results and the comparison of routing protocols indicate that the bio-inspired routing algorithm, namely AntHocNet, is a viable option for FANET routing. Moreover, the factors affecting the connectivity of nodes are investigated. The results indicate that there is a minimum threshold value for the average

number of neighbors to form a non-partitioned network, and this minimum value decreases as the node speed increases.

Keywords: Flying Ad-Hoc Networks, Swarm Intelligence Algorithms, AntHocNet, Routing Algorithms, Fixed-Wing UAV

ÖZ

SABİT KANATLI İNSANSIZ HAVA ARAÇLARININ OLUŞTURDUĞU UÇAN DÜZENSİZ OLUŞUMLU AĞLAR İÇİN BIYOLOJIK ESİNLENMELİ VE GELENEKSEL YÖNLENDİRME ALGORİTMALARININ KARŞILAŞTIRILMASI

Şeker, Burak
Yüksek Lisans, Bilişim Sistemleri Bölümü
Tez Yöneticisi: Doç. Dr. Altan Koçyiğit
Ortak Tez Yöneticisi: Doç. Dr. Enver Ever

Eylül 2021, 139 sayfa

Sabit Kanatlı Uçan Düzensiz Oluşumlu Ağlar, ağı oluşturan elemanların sabit kanatlı insansız hava araçları veya dronlar olduğu ve bu elemanların 3 boyutlu ortam içerisinde dağıldığı iletişim ağlarıdır. Sabit Kanatlı Uçan Düzensiz Oluşumlu Ağlarda ağ topolojisi, ağ elemanlarının 3 boyutlu ortamdaki hareketliliği nedeniyle çok dinamiktir. Bu nedenle, yönlendirme algoritmasının seçimi ana zorluklardan biridir. Biyolojik esinlenmeli AntHocNet yönlendirme algoritması, diğer alternatifler arasında, Sabit Kanatlı Uçan Düzensiz Oluşumlu Ağların sorunlarıyla başa çıkmada ciddi bir potansiyel sunar. Bu tezde, NS3 ağ simülasyon ortamı kullanılarak biyolojik esinlenmeli bir yönlendirme algoritması olan AntHocNet ve geleneksel yönlendirme algoritmaları olan AODV ve OLSR, Sabit Kanatlı Uçan Düzensiz Oluşumlu Ağ uygulamaları için karşılaştırılmıştır. Bu karşılaştırmada, artan topoloji değişim hızının ve artan ağ yükünün, karşılaştırılan yönlendirme algoritmalarının performansını nasıl etkilediği incelenmektedir. Sonuçların analizi ve yönlendirme protokollerinin karşılaştırılması, biyolojik esinlenmeli AntHocNet yönlendirme algoritmasının Sabit Kanatlı Uçan Düzensiz Oluşumlu

Ağlarda yönlendirme algoritması olarak uygun bir seçenek olduğunu göstermektedir. Ayrıca, ağ elemanlarının birbirlerine bağlanabilirliklerini etkileyen faktörler araştırılmıştır. Sonuçlar, bölümlenmemiş bir ağ oluşturmak için ortalama komşu sayısı için minimum bir eşik değerinin olduğunu ve düğüm hızı arttıkça bu minimum değerin azaldığını göstermektedir.

Anahtar Kelimeler: Uçan Düzensiz Oluşumlu Ağlar, Sürü Zeka Algoritmaları, AntHocNet, Yönlendirme Algoritmaları, Sabit Kanatlı İnsansız Haca Aracı

To my loving family

Ayşe Begümhan Şeker, Alptuğ Şeker, Nurten Şeker, Nedim Şeker

ACKNOWLEDGMENTS

I would like to thank my supervisor, Assoc. Prof. Altan Koçyiğit, and my co-supervisor, Assoc. Prof. Enver Ever, for their time, effort, and guidance. Altan Koçyiğit is a great academician, and I learned much from him. He contributed to every aspect of my life. As well as Altan Koçyiğit, Enver Ever also provided me with a lot of insight and contributed to my thesis and me as a human being. It was an honor to work with them.

Being a graduate student at METU and an embedded software engineer at TÜBİTAK-SAGE was a tough challenge. It makes 'the time' the most scarce resource, and it makes you beg for it. In these three tough years, my wife Ayşe Begümhan ŞEKER continuously supported me at every point and gave me the necessary motivation and courage to finish my Master's degree. I want to provide her with my special thanks. I also would like to thank my family, Nedim ŞEKER, Nurten ŞEKER, and Alptuğ ŞEKER, who always supported me in my journey. I appreciate their support and understanding very much.

And finally, I want to thank my colleagues and supervisors at TUBİTAK-SAGE for their consideration, support, and understanding.

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

LIST OF ABBREVIATIONS

ACO	Ant Colony Optimization
AODV	Adhoc On-demand Distance Vector (Routing Protocol)
AOMDV	Adhoc On-demand Multipath Distance Vector (Routing Protocol)
ANoN	Average Number of Neighbors
AWSN	Adhoc Wireless Sensor Network
CDMA	Code Division Multiple Access
CLMN	Column (Mobility Model)
DSDV	Destination-Sequenced Distance Vector (Routing Protocol)
DSR	Dynamic Source Routing
DTN	Delay-Tolerant Network
EGM	Enhanced Gaussian-Markov (Mobility Model)
FANET	Flying Adhoc Network
FW-UAV	Fixed Wing Unmanned Arial Vehicle
GM	Gaussian-Markov (Mobility Model)
GRP	Geographic Routing Protocol
GPRS	General Packet Radio Service
GSM	Global System For Mobile Communications
HWMP	Hybrid Wireless Mesh Protocol
LTE	Long-term Evolution
MANET	Mobile Adhoc Network
MG	Manhattan-Grid (Mobility Model)
MID	Multiple Interface Declaration

MPR	Multi-point Relay
NC	Nomadic Community (Mobility Model)
OLSR	Optimized Link State Routing
PBA	Proactive Backward Ant
PFWA	Proactive Forward Ant
PLR	Packet Loss Ratio
PPRZM	Paparazzi Mobility Model
PSMM	Particle Swarm Mobility Model
RD	Random Direction (Mobility Model)
RBA	Reactive Backward Ant
RFWA	Reactive Forward Ant
RGR	Reactive-Greedy-Reactive (Routing Protocol)
RPGM	Reference Point Group Mobility
RERR	Route Error
RREQ	Route Request
RREP	Route Reply
RW-UAV	Rotary Wing Unmanned Arial Vehicle
RWP	Random Waypoint (Mobility Model)
QoS	Quality of Service
SATCOM	Satellite Communication
SI	Swarm Intelligence
ST	Smooth Turn (Mobilty Model)
TC	Topology Control
TDMA	Time Division Multiple Access
TxR	Transmission Range
U2U	UAV to UAV

U2G	UAV to Ground
UAV	Unmanned Arial Vehicle
UDP	User Datagram Protocol
VANET	Vehicular Adhoc Network
WSN	Wireless Sensor Network

CHAPTER 1

INTRODUCTION

Technology is evolving toward working together to achieve significant tasks. This technology trend spreads over many research areas, from BigData clusters to swarm robotic applications. The advancements of silicon technologies that make the sensors and the computing power of integrated chips improve each passing year while becoming cheaper strongly support this trend. In this new age of *Technological Democracy*, where individual agents become more intelligent and more affordable, new research areas concerning the swarm of unmanned areal vehicles (UAVs) getting more and more attention from researchers [1] because the use of multiple UAVs makes the completion of the missions easier by providing flexibility and robustness in the task assignments [2].

The swarm of UAVs has a great potential to provide many advantages in completing tasks or providing solutions for many application areas. However, this also brings some challenges. One of the main challenges is the issue of mobility. UAVs are much faster than the nodes of other ad hoc networks, namely MANETs and FANETs. Moreover, the movement patterns of UAVs are 3D movements while the nodes of MANETs and VANETs move in a 2D environment. Due to this, the mobility models used for UAVs differ from the mobility models used for the nodes of MANETs and VANETs. Also, the network topology of FANETs is more dynamic than other ad hoc networks.

The interest in the swarm of UAVs also brings communication challenges with it, which is understandable. After all, to work together, the agents of any crowd need to communicate with each other. Moreover, since FANET network topology is more dynamic than MANET and VANET, it requires special solutions to cope with the higher rate of change in network topol-

ogy. Accordingly, the need for communication between UAVs, which provides many tough challenges for researchers, is acknowledged and makes Flying ad hoc Networks (FANETs) an important area of research.

As chips' performance, scale, and functionality are improving day by day [3] and wireless technology, which binds the agents of the swarm of UAVs, are getting more capable, the need for communication increases with the new requirements expected from FANET applications. The increase in communication requirements further increases the significance of the optimization of network communication.

Routing protocols, or algorithms, have been sharing the soul of this age of *Technological Democracy* for some years. Especially in the last decade, Swarm Intelligence (SI) based, or bio-inspired, routing algorithms targeting ad hoc network applications are becoming popular among researchers because of the innate similarity between SI-based algorithms and ad hoc networks. SI-based algorithms, inspired by the different swarms of life forms, such as ants, bees, birds, and wolves, also carry the properties of *autonomous agents*, *self-organization*, and *division of labor*. These properties make bio-inspired routing algorithms a good match for ad hoc network applications. Because of the innate similarity between bio-inspired routing algorithms and ad hoc networks, many researchers have offered bio-inspired algorithms for ad hoc networks for the last two decades [4], [5], [6].

Even though bio-inspired algorithms are used for ad hoc networks in recent years, there is a gap within the literature concerning bio-inspired routing algorithms in FANET Applications consisting of Fixed-Wing UAVs (FW-UAVs). The literature lacks the research on the performance of bio-inspired routing algorithms in FW-UAV FANET applications with appropriate mobility models and using 3D network simulations. In order to fill this gap, this research compares bio-inspired and traditional routing algorithm performances and observes the effect of mobility model and node speed, in other words, the effect of the topology change rate, on the ad hoc network communication performance of routing algorithms.

1.1 Problem Statement

The problem of routing the network traffic in FW-UAV networks is a significant problem and getting more and more popular. Establishing efficient communication among the nodes within a FANET application is crucial to make the swarm of nodes complete their assigned tasks effectively. Even though bio-inspired routing algorithms are likely to fit well to FANET applications and have the potential to provide a promising solution for routing in FANET applications, there is a lack of research that shows the applicability of bio-inspired routing algorithms to FANET applications.

This research addresses the problem of routing in FW-UAV FANETs by comparing the bio-inspired routing protocols with traditional routing protocols in a 3D space and measuring the effect of various mobility models and various network loads on the performance of the compared routing algorithms.

Apart from routing, the issue of the minimum node density, i.e., the minimum average number of neighbors, to enable full connectivity is also addressed by this study because determining the average number of neighbors is necessary for designing an ad hoc network application. Since the nodes are mobile in FANETs, the effect of the node speed on the minimum value of the average number of neighbors for providing a non-partitioned network is observed within the scope of this research.

1.2 Purpose Statement

The purpose of this experimental research is to inquire whether a bio-inspired routing algorithm, namely AntHocNet, is a viable option for FW-UAV Networks. The network performance of the above-mentioned bio-inspired routing algorithm and the network performances of the traditional routing algorithms, namely AODV and OLSR routing algorithms, are compared to carry out this inquiry. Executing network simulations for FW-UAV ad hoc networks in a 3D simulation environment and measuring mean end-to-end delay, mean end-to-end jitter, packet loss ratio, mean hop count, and routing overhead parameters provide a base for comparing routing protocols.

The nature of FW-FANETs requires the simulations to include scenarios where the speed of nodes is much higher than scenarios used for MANETs or VANETs. Moreover, apart from the high node speed requirement, the usage of mobility models suitable for FW-FANETs is also required. Following these, the scenarios determined for this study aim to scan various node speeds using various FW-UAV mobility models, such as PPRZM and EGM.

1.3 Significance of Study

It is essential to show that bio-inspired routing algorithms can perform better than traditional routing algorithms because there is a gap within the literature concerning the use of bio-inspired routing algorithms in FW-UAV ad hoc networks. Moreover, within the literature, most of the studies concerning routing lack the necessary simulation environment to address the routing issue of FW-FANET networks, such as 3D environment, the usage of an appropriate mobility model, and the spanning the appropriate node speed ranges. This research aims to fill this gap.

In this respect, one of the contributions of this research is that it compares network performances of a bio-inspired routing algorithm, namely AntHocnet, and the traditional routing algorithms, namely AODV and OLSR, by conducting simulations in a 3D environment and with mobility models offered for FW-FANETs.

The second contribution of this research is that it compares the mobility models designed for FW-FANETs, such as PPRZM and EGM, with RWP, which is the mobility model used in most ad hoc network researches within the literature. In other words, this research investigates the effect of the mobility model on the performance of the mentioned routing algorithms. Comparison of PPRZM and EGM with RWP also provides a base for comparing the results of this study with other researches concerning ad hoc network routing since most of the studies use the RWP mobility model in their simulations.

In [7], it is stated that to obtain a non-partitioned network communication, in other words, an above 90% *connected coverage*, there is a minimum value regarding the average number of neighbors in immobile ad hoc networks. If the value of the average number of neighbors drops below a certain threshold, then the network becomes partitioned and the connectivity

of nodes degraded. Therefore, to execute a simulation scenario where the network stays non-partitioned, the effect of node speed on the minimum value of the average number of neighbors is investigated. In other words, it can be said that another contribution of this research is about the effect of the node speed on the minimum value of the average number of neighbors, which is required for a non-partitioned network communication; and, within the boundaries of this study, how the minimum value of the average number of neighbors changes with increasing node speed is examined.

1.4 Research Questions and Hypotheses

The research questions and hypotheses of this study are as follows:

1. How does an increase in the topology change rate affect the communication performance of the compared routing algorithms, namely AODV, OLSR, and AntHocNet?

    - It is anticipated that the communication performances of the compared routing algorithms, i.e., AODV, OLSR, and AntHocNet, degrade as the topology change rate increases because it gets harder to execute route maintenance processes. We expect to see that the communication performance provided by AntHocNet is better in increasing topology change rates than the communication performances provided by AODV and OLSR.

2. How does an increase in network load affect the communication performance of the compared routing algorithms, namely AODV, OLSR, and AntHocNet?

    - It is anticipated that communication performances of the compared routing algorithms, i.e., AODV, OLSR, and AntHocNet, degrade as the network load increases because of the network congestion. We expect to see that the communication performance provided by AntHocNet is better in various network loads than the communication performances provided by AODV and OLSR.

3. What factors affect the connectivity of the whole network in an FW-UAV FANET application?

- We expect to see that the average number of neighbors affects the connectivity of the nodes within the FW-FANETs, if not appropriate causing a partitioned network.

1.5 Assumptions

The assumptions of this research are stated as follows:

1. The data collected with the NS-3 simulation tool provides a good base for finding the answers to the research questions of this study.
2. It is assumed that when the packet loss ratio is below 0.1, the network is non-partitioned.
3. It is assumed that IEEE 802.11g technology is used in FANETs.
4. It is assumed that the transmission range is adjustable.
5. It is assumed that within the transmission range, RSSI is excellent, and there is no free space loss for signal power, while outside of the transmission range, no signal can be received.
6. It is assumed that there are no obstacles between nodes or UAVs.
7. It is assumed that the range of speed for FW-UAVs is between 5 m/s to 120 m/s.

1.6 Thesis Organization

This study consists of six chapters. The introduction chapter presents the problem statement, statement of purpose, and statement of significance. Moreover, the first chapter provides the formulations of research questions, hypotheses, and assumptions. The background information chapter, i.e., chapter two, provides brief information regarding the wireless ad hoc networks, routing protocols, and mobility models for ad hoc networks. The second chapter also explains the wireless ad hoc networks, routing protocols, and mobility models. The third chapter provides a literature review referenced by this research. The fourth chapter

presents the research methodology in detail by explaining the simulation environment and how this study uses the simulations by providing the dependent and independent parameters and constants and simulation scenarios used for data collection. The fifth chapter analyzes the simulation results. Finally, the last chapter displays the significant results, the contributions of this study, the limitation of this research, and future recommendations.

CHAPTER 2

BACKGROUND INFORMATION

This chapter presents background information concerning ad hoc networks, routing protocols, and Swarm Intelligence. Moreover, The types of ad hoc networks are briefly discussed together with their essential aspects. This chapter also discusses the types of mobility models, their fundamental aspects, and their effect on network communication. Within the lines of this presentation, it is also intended to answer why ad hoc networks need unique routing mechanisms. Together with this inquiry, these routing mechanisms are categorized, and the state of art routing algorithms is briefly explained for completeness. Finally, in this chapter, brief information regarding swarm intelligence and swarm intelligence-based routing algorithms are given. Although this work does not intend to put a complete taxonomy of bio-inspired or swarm intelligence-based routing algorithms, the key aspects of this branch of routing algorithms and their place in ad hoc routing protocols are shown.

2.1 Wireless Ad hoc Networks

Ad hoc networks are the type of networks that are established without infrastructure or a predetermined topology [8]. Ad hoc networks are established between nodes, which are the communicating components of the network, on the fly and according to the communication need. In other words, this kind of network does not have 'fixed routers' as Jayakumar puts it and consequently does not have predetermined routing paths [8]. The packet routing is determined according to the flows and network traffic and without a centralized control mechanism. Since there is no centralized network control mechanism, this responsibility is divided among the nodes. In other words, the nodes are not just the end-users but also responsible for routing

the packets for their neighbors and responsible for determining where to relay packets. This aspect of ad-hoc networks implies that the packets travel from node to node until they reach their destinations or are lost along the way. For this reason, an ad hoc network is also named as *multi-hop network*.

Ad hoc networks have a dynamic topology, which implies that the network's topology is changing throughout the network's lifetime. This characteristic may stem from different reasons depending on the type of the ad hoc network. For instance, in Wireless Sensor Networks, some nodes may disconnect because of some environmental effect or battery run out and cannot continue to route packets for their neighbors or themselves. In such cases, the traffic should be rerouted because the path using ex-node is unavailable, and therefore, the topology is changed. On the other hand, for mobile ad-hoc networks, the topology of nodes changes due to mobility. Also, the set of their neighbor nodes changes as a node moves. This situation may necessitate updating the existing network traffic routes. In this changing network topology, the nodes should be smart enough to cope with change, and they must implement some kind of routing algorithm that steers the network traffic according to the situation.

Another point that should be mentioned about ad hoc networks is that they are generally designed for a specific purpose or a specific use case. According to the purpose of design;

- the network can consist of homogeneous or heterogeneous nodes,
- the nodes can be clustered or not,
- nodes can be static or mobile,
- number of nodes, deployment area, and network traffic may vary,
- and hence the use of different routing algorithms may be needed.

Within the literature, ad hoc networks are generally categorized according to the mobility type of nodes. The most common categories are WSNs, MANETs, and VANETs. However, with the increasing interest in unmanned aerial vehicles (UAVs), another category of ad hoc networks, namely FANETs, is getting more and more attention. The properties, challenges, and differences of these ad hoc networks are examined in the following sections.

2.1.1 Wireless Sensor Networks

Wireless Sensor Networks (WSNs), or ad hoc Wireless Sensor Networks (AWSNs), are generally designed to monitor the events within target environments. However, according to the aimed application, the nodes in WSN applications need to have computing and data processing capabilities apart from sensing environmental changes [9]. In this kind of ad hoc network, the nodes are generally stationary and scattered around the deployment area. The positions of nodes, or sensors, can be arbitrary and might not be determined before deployment [10]. In other words, wireless sensor nodes are deployed *'in ad hoc manner without any sort of appropriate planning or research'* [6]. This nature requires a communication structure that can be applied for various network topologies because, after nodes are deployed, they need to establish communication among themselves and start transmitting data to a base station or sink node. For this communication structure to operate, the nodes of the Wireless Sensor Network need to have the ability to self-organize autonomously [10].

Another challenge that requires specially designed routing algorithms for WSN is the issue of power consumption. In WSNs, the nodes are generally designed to be cheap devices that can sustain the target environment independently because they are generally deployed in large numbers. This situation makes power consumption a crucial issue; because it is close to impossible or impractical to change the batteries of nodes or recharge them [6].

For a sensor node, the main power-consuming operation is wireless transmission; because it is costly to transmit wireless signals. For this reason, in WSN applications, sensor nodes are designed to have a limited transmission range to reduce power consumption. Since the sensors have a limited transmission range, all sensor nodes may not transmit directly to the base station or a subset of other nodes. Hence, sensor data is relayed via neighbor nodes until they reach their destinations. Furthermore, the nodes may die, in other words, leave the network. There may be many reasons that cause a node to leave the network. For instance, it is possible for a node to be damaged, or destroyed, due to environmental effects. It is also possible for the battery of a node to dry up. Either way, the topology of the network may change because due to a node leaving the network. In order to organize the mentioned multi-hop network mechanism in changing network topologies autonomously, specific routing algorithms are needed. The routing algorithm implemented on each node could also be designed to prolong

network lifetime by balancing the power consumption of nodes achieved by balancing the packet forwarding responsibility among nodes.

2.1.2 Mobile Ad Hoc Networks (MANETs)

MANETs are one of the most popular and most researched branches of ad hoc networks. A MANET consists of mobile nodes, which have wireless communication capabilities. These mobile nodes can set up a communication network within themselves without any *centralized administration* [11]. Since nodes are mobile, they quickly join or leave the network, and thus the network formed between these nodes has a changing topology [4].

In MANETs, the mobility of nodes is generally minimal and does not change drastically. Even though they move randomly, the node speeds are generally between 1 m/s to 10 m/s. Moreover, the nodes are mobile on a plane, i.e., a 2D space [12]. Compared to VANETs and FANETs, MANETs' topology change rates are lower. However, they have higher energy constraints than FANETs, and VANETs [13]. Moreover, Mobile Ad Hoc Networks generally have higher densities as MANET applications are designed for smaller areas than areas covered in FANET and VANET applications.

2.1.3 Vehicular Ad Hoc Networks (VANETs)

VANETs are a subset of ad hoc networks and consist of vehicles, such as buses, cars, or trucks, as nodes. It is a crucial technology for *intelligent transportation systems* and the vision of *smart city* [14]. The main idea behind VANETs is the communication of vehicles that exchange information among themselves and with transportation infrastructure to provide traffic management, safety, efficiency, and quality and commercial applications.

The communication need of VANETs can be divided into two main categories, namely, communication among the vehicles, i.e., moving nodes, and the communication with transportation infrastructure, i.e., *roadside units (RSU)*, which have access to the Cloud and handle the data storage and analysis responsibilities [15].

The network topology of a VANET is more inclined to change than a MANET's network topology; because the mobility degree of a VANET is higher than that of a MANET, and nodes move at higher speeds ranging from 5 m/s to 25 m/s. However, while in MANETs, the node mobilities are not necessarily confined to particular regions and are usually modeled randomly, in VANETs, vehicles moving upon the roads and the mobility models used for VANETs are less random than MANETs. Therefore, the mobility models used for VANETs are not pure random models. In VANETs, mobility models such as *Manhattan Grid*, which uses grid road topology on a 2D plane, or other prediction-based models are considered [12], [13].

The difference in nodes and deployment environment also brings some variations in energy constraints and node density. In VANETs, energy consumption is not considered a critical issue because vehicles can provide enough power, while in MANETs, it is one of the main issues because power is a scarce resource for MANET nodes that have scarce resources. Moreover, VANET applications are designed to cover much larger areas than that of MANETs, making the node density lower than the node densities in MANET applications.

2.1.4 Flying Ad Hoc Networks (FANETs)

FANETs are another subset of ad hoc networks that caught much attention within the last decade [16], [17], [18], [19], [20]. A FANET consists of unmanned aerial vehicles (UAVs) as nodes, and similar to other ad hoc networks, the communication paths are established in an ad hoc manner between the nodes.

The application areas of FANETs are vast, from monitoring and surveillance to military search and destroy missions or even search and rescue missions in disaster areas [2]. These application areas need different classes of UAVs; therefore, various kinds of UAVs may be used according to the aimed application area. In some applications, the UAVs need to move faster and longer than others, and thus Fixed-Wing UAVs (FW-UAV) are used [21]. In some other applications, UAVs need to place themselves almost stationary above the targeted area, for instance, around a disaster area; therefore, Rotary-Wing UAVs (RW-UAVs) are selected [21],

[22]. While FW-UAVs are more suitable for applications intended for larger operation areas, other applications aimed for smaller areas are a better fit for RW-UAVs [23].

In any case, FANET nodes are a lot faster than MANETs, and generally faster than VANETs. For RW-UAVs, the node speeds are around 15 m/s - 20 m/s [12]. For FW-UAVs, on the other hand, nodes may move with a velocity of 30 m/s up to 120 m/s [12], [15]. Another important point that differentiates FANETs from VANETs and MANETs is the node mobility type. While nodes generally move in a 2D space in MANETs and VANETs, FANETs have a 3D mobility because the nodes are UAVs. This situation makes designing, simulating, and verifying FANET applications more challenging. Another point that should be mentioned is the issue of mobility models. While it is appropriate to use randomized mobility models for RW-UAVs such as 3D Random Way Point (RWP) mobility model [12]; Random Way Point (RWP), Random Direction (RD), or Manhattan mobility models are not suitable for FW-UAVs due to the movement characteristics [23].

The properties of FANETs mentioned above, i.e., higher node speeds and 3D mobility, make the network topology a lot more dynamic than MANETs and VANETs. On the other hand, the node density of a FANET application is usually lower than the node densities of MANET and VANET applications.

2.2 Routing Protocols In Ad Hoc Networks

As explained in Section 2.1, various kinds of ad hoc networks have various requirements to be satisfied. The discussions provided identifies the management of the data flow, i.e., routing, as a common issue for the existing studies on ad hoc networks. After all, the problem of routing in ad hoc networks is an NP-hard optimization problem [4]; which means that in the worst case, finding a solution to the question of how to route a packet most efficiently requires exponentially increasing time as the size of the network increases [24].

In order to overcome the problem of delivering network packets in ad hoc networks, various routing protocols are offered in the literature. This section is intended to present not a complete taxonomy of routing protocols but the main branches. In other words, in this section,

the selected traditional routing algorithms are presented and explained to provide a base for this comprehensive work.

The main distinction presented in the literature between routing algorithms is based on when to determine and how to maintain a route for a particular transmission. In proactive routing algorithms, for instance, firstly, routing messages are exchanged among nodes to produce a snapshot of the current topology. In other words, each node obtains the information of the set of neighbor nodes and the link qualities. Then, when a packet is needed to be sent to a particular node, this information is used to select the source-to-destination route. In summary, the information for route selection is proactively generated before it is needed. In reactive routing algorithms, on the other hand, when a packet is sent to a particular node, control messages are sent from the source towards the destination to find a source-to-destination path. According to the reply messages, the route selection occurs, and the packet is sent. In reactive routing algorithms, the node discovers the available routes by reacting to the need for packet transmission.

There are also other types of routing algorithms, such as hybrid algorithms, which integrate both reactive and proactive routing algorithm components [13], [25], or bio-inspired, i.e., Swarm Intelligence (SI) based, routing algorithms [4], [26], [27], [9].

In the following subsections, some of the traditional routing algorithms are explained in more detail. The selected algorithms are OLSR for proactive routing algorithms and AODV for reactive routing algorithms. Moreover, brief information concerning swarm intelligence-based algorithms is presented, and AntHocNet as a bio-inspired routing algorithm is explained.

2.2.1 Optimized Link State Routing (OLSR)

OLSR is a table-driven, proactive routing algorithm that is widely used primarily for MANET applications. It is a proactive routing protocol in which every node holds all necessary routing information regarding other nodes before there is a need for packet transmission [28].

Each node within the network selects its MPR (Multipoint Relay) nodes. MPRs are a subset of neighbor nodes, and they are the relay point for a particular node for 'flooding control

traffic' [29]. Only the MPRs of a specific node re-transmit the broadcast control messages to the network. A particular node selects its MPR nodes to relay a message to all of its two-hop neighbors via selected MPR nodes. This mechanism increases the efficiency in flooding control messages and reduces routing overhead by cutting redundant control messages.

According to the RFC 3626, OLSR uses three fundamental types of messages, namely *Hello* messages, *Topology Control (TC)* messages and *Multiple Interface Declaration (MID)* messages [29]. *Hello* messages are periodic messages which are broadcasted by nodes to their 1-hop neighbors. With *Hello* messages, nodes periodically learn about their neighbor nodes, available links and update their routing tables accordingly. *TC* messages are used to keep track of the topology. With *TC* messages, nodes can hold the most recent information regarding the link states. Also, *TC* messages hold information of MPR selectors of the sender node, which indicates the set of nodes whose routing messages can be acquired through this sender node due to that the sender node will re-transmit them. *MID* messages are used to inform the network about multiple interfaces. When a particular node has multiple interfaces through which communication can be established, it periodically advertises this information via its MPRs using MID messages.

In OLSR, thanks to periodic *Hello* and *TC* messages, route selection does not cause additional delay for data flow since the routes are available all the time [30]. This property makes OLSR a suitable candidate for delay-sensitive applications. Moreover, OLSR can adapt to the changes in the topology and is suitable for ad hoc networks. However, there is a balance and a limit to how much OLSR can adapt.

OLSR uses periodic messages to flood MPRs in order to detect the topology changes. If it is intended to use OLSR in highly dynamic topologies, control message intervals should be reduced to adapt to the topology changes. However, reducing the periodic message intervals increases the control traffic throughout the network and consequently increases routing overhead significantly. In other words, the proactive nature of the OLSR routing algorithm backfires as the topology change rate increases. Because, as the topology change rate increases, it gets harder to learn the topology beforehand.

Regardless, it is one of the traditional routing algorithms widely used in ad hoc network applications.

2.2.2 Adhoc On-Demand Distance Vector (AODV)

AODV is one of the most studied reactive routing protocols, which mainly focus on handling dynamic topologies. AODV is an on-demand protocol that indicates that the routes are established when a node, i.e., a source node, needs to send data to another node, i.e., destination node, and this route information is held by nodes as long as the source node needs them [31]. Another property of AODV is that it requires less processing and memory resources while being able to cope with dynamic link changes [28], [32]. AODV can notice the link breaks and spread this information among nodes to update their routing tables accordingly.

According to RFC 3561, AODV uses three fundamental control message types to provide above mentioned utilities, namely *Route Request (RREQ)*, *Route Reply (RREP)* and *Route Error (RERR)*. For circulating the control messages, AODV uses UDP/IP stack [32]. As mentioned above, the AODV is an on-demand protocol, meaning that it starts on its own accord when a packet needs to be sent by a node, and nodes do not know the route to send the packet. At this point, the source node broadcasts an RREQ message, which carries the information of the destination node along with some sequence numbers, which is used to avoid routing loops and stale route information and indicates the *freshness* of the information. Each node that receives the RREQ message controls its routing tables to provide a route for the request. If a suitable and fresh route is found, an RREP message is sent back to the source node, indicating that the route it sought is available via the node that sends back the RREP message. If a suitable route is not found, the RREQ message is forwarded until a route is found. In other words, RREQ messages are flooded towards the destination node if no valid route information toward the destination node is present among intermediate nodes. Furthermore, RREP messages are unicasted back via the freshest and the shortest route towards the source node.

In AODV, route lengths are considered in the scale of the number of hops. After receiving an RREP and starting transmission, the source node can receive another RREP with a shorter

route. In this case, the source node can update its routing table and use the better, i.e., shorter, route to send data packets [30].

The nodes hold the active route information in their routing tables as long as the route is used. The nodes determine whether a route is active or not according to a timeout counter. If a particular route in the node's routing table is not used for a predetermined time interval, that route is considered inactive and deleted from the node's routing table. In other words, nodes only keep information about active routes. Different from RREQ and RREP messages, RERR messages are used to adapt to changes in topology. Whenever a node detects a broken link that is used by the active routes that are in the routing table, it unicasts a RERR message to the source node of the concerned active route in order to inform it that the route is broken so that the source node initiates an RREQ message to set up a new route [30]. By using the RERR messages, AODV can adapt to changes in the network topology.

In AODV, thanks to its reactive or on-demand nature, it is not necessary to send periodic control messages within the network, which decreases the routing overhead. AODV is also highly scalable and a low-cost routing algorithm that can adapt to changes in the network. However, it should be emphasized that whenever a new transmission is needed, it puts some initial delay until the route is established. Moreover, the route breaks increase as the topology changes faster. Consequently, delay and routing overhead increase. Nevertheless, it is another traditional routing protocol widely studied and used in ad hoc networks.

2.2.3 Swarm Intelligence Based Routing Algorithms

SI is inspired by the group behavior of various life forms such as ants, bees, birds. The common notion that can be derived by inspecting the life of these organisms is that even though they form a group from individual agents with limited capability, the group itself can accomplish its purpose efficiently. Inspired by the fact that the group of agents with limited capability completes complex tasks efficiently, bio-inspired algorithms are investigated in studies such as [33], [34], [35]. These studies interest simple agents that communicate with neighbor agents within a complex environment to realize a complex job, like an ant colony or a flock of birds [36].

The main characteristics of a swarm intelligence behavior are *self-organization* and *division of labor* [5]. Self-organization refers to the ability to execute necessary jobs by using feedback mechanisms without any centralized control or planning. Moreover, the division of labor indicates the property of a group where different kinds of responsibilities are divided among agents, consequently differentiating the agents within the swarm.

These characteristics make bio-inspired algorithms flexible and efficient in solving nonlinear problems [37], such as routing issues in ad hoc networks like WSN and MANETs [5]. In recent years, SI-based routing algorithms are also considered for FANETs, as they also hold great potential to solve the issue of routing in highly dynamic network topologies [38], [39], [40].

In [37], swarm intelligence algorithms are divided into eight different categories, namely *Ant Algorithms*, *Bee Algorithms*, *Bat Algorithms*, *Particle Swarm Optimization*, *Firefly Algorithm*, *Cuckoo Search*, *Flower Pollination Algorithm* and *Other Algorithms*. Each of these groups gets themselves a fair amount of research area within the literature, and some are applied for routing in ad hoc networks.

It is not in the scope of this research to put a complete taxonomy of swarm intelligence-based algorithms. However, the AntHocNet routing algorithm, which belongs to the family of Ant Algorithms, is presented in the following section because it is claimed to be superior to AODV and OLSR for FW-UAV networks in this research.

2.2.3.1 Ant Algorithms and Ant Colony Optimization

AntHocNet belongs to the category of Ant Algorithm and employs an approach similar to Ant Colony Optimization (ACO) [41]. For this very reason, firstly, how the ACO works is investigated. Then, the way the AntHocNet routing algorithm uses the ACO is clarified while explaining the working mechanism of the AntHocNet as a routing protocol. This section also intends to emphasize how swarm intelligence behavior is adapted and used for routing algorithms in the example of the AntHocNet routing algorithm.

Ant Algorithms, and also ACO and AntHocNet, are inspired by the foraging mechanism of ants [37], [24]. Ants cooperate to find the shortest path from their nest to the food source. This cooperation is conducted by exchanging messages that all other ants can understand within the colony, namely *pheromone*. The *pheromone* is a chemical that can be spilled and smelled by ants [24].

If an ant finds a food source, it carries it back to its nest while leaving a trail of pheromones. Other ants that pick the trail also start transferring the food to the nest. Pheromone is a kind of chemical that evaporates with time, and therefore the density of the *pheromone* decreases with time. If there are multiple trails to the food source, ants choose the path that the pheromone density is higher than others. The longer the path, the lower the pheromone density within the trail and the fewer ants choose. This mechanism enforces the shortest path to the food source, and with time, all the ants start to use the shortest path [36], [24], [37].

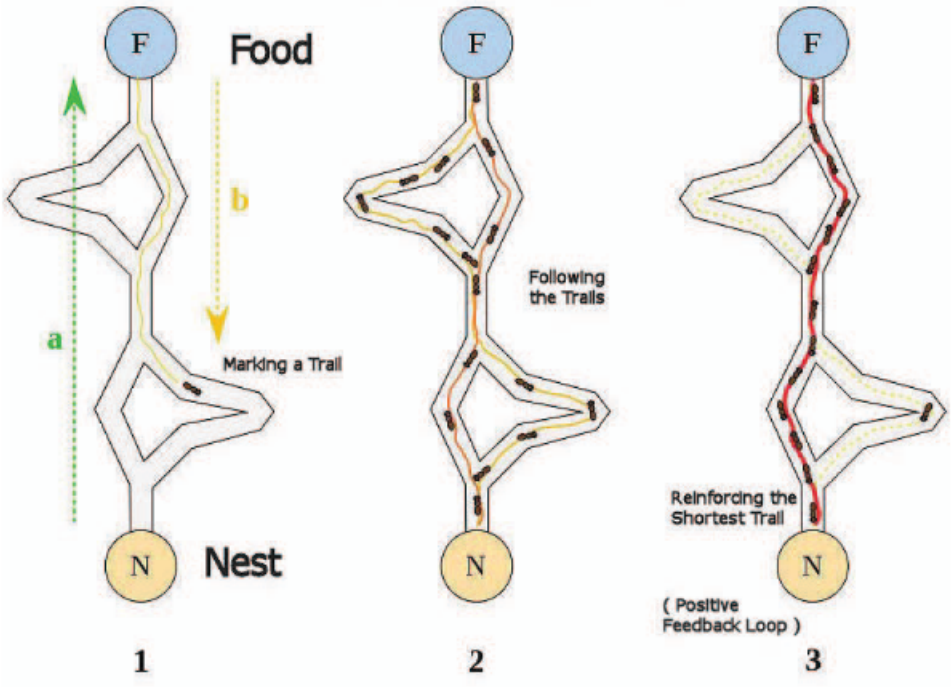


Figure 1: Ant Colony Optimization Mechanism [38]

The above-mentioned meta-heuristic approach is used in ACO algorithms. In ACO, artificial ants follow a stochastic process to construct a solution iteratively for the search. As the partial solutions pile up, more information is ready to be taken into account. In [24], it is stated that the information created and accumulated in each iteration is the *heuristic information*

that is obtained from the iteration being solved and the *artificial pheromone trail* that is an indicator of agents' search experience. As these iterations are being made, the solution that the algorithm forms usually converge to an optimal solution [37]. The use of a stochastic process provides an ACO algorithm to try a broader range of possible solutions than greedy approaches [24].

2.2.3.2 AntHocNet Routing Algorithm

As stated, AntHocNet benefits from the ACO approach and applies it to routing in MANETs, which is an NP-hard problem. AntHocNet contains both reactive and proactive components, which makes it a hybrid algorithm. The reactive component of AntHocNet is that it tries to find solutions when a data packet is needed to be sent. In other words, just like AODV, it finds routes only when active communication is requested. On the other hand, AntHocNet has a proactive component which is that even though a route is obtained for the packet transmission, AntHocNet tries to find better routes and tries to maintain the current solution due to its iterative nature [41].

In the AntHocNet routing algorithm, ants described in ACO turn into control messages that carry route information and route quality information, i.e., pheromone. There are five different control message types present in the AntHocNet algorithm, namely, *reactive forward ants (RFWA)*, *reactive backward ants (RBA)*, *proactive forward ants (PFWA)*, *proactive backward ants (PBA)* and *pheromone diffusion (PD)* messages.

When data transmission is needed, the source node initiates *reactive route setup process* [41]. In this process, the source node sends an RFWA to its neighbors. If a node, sending or forwarding an RFWA message, has information about the destination of RFWA in its pheromone table, then it unicasts the RFWA message using the information that resides in its pheromone table; and if it does not have this information, then it broadcasts the RFWA message. RFWA message travels from the source to the destination while carrying the node chain information that has traversed along the way. After the RFWA message reaches the destination, it is converted into an RBA message and backtracks to the source node. RBA message updates

the pheromone tables as it travels back to source node according to the link qualities. The pheromone tables are the accumulation place of the solutions, i.e., better routes.

In this *reactive route setup process*, some additional points should be emphasized. First of all, when intermediate nodes are unicasting an RFWA message, the destination is selected probabilistically and according to the pheromone information in the node's pheromone table. This mechanism provides AntHocNet to find other possible solutions as well. Secondly, according to the implementation of the algorithm, intermediate nodes can forward only the first RFWA message or up to a certain number. However, Ducatelle suggests that the forwarding nodes should discard other RFWA messages after forwarding the first one to reduce the routing overhead [41]. Finally, when an RFWA message reaches its destination and is converted into an RBA, other RFWA messages that arrived at the destination after the first one will be discarded. In *reactive route setup process*, the primary purpose is finding a route (optimal or not) that the data transmission can be executed and establishing a communication session.

After the communication session is established, *proactive route maintenance process* starts, and it is executed as long as the communication session is alive. This process contains two different parallel jobs, namely *Pheromone Diffusion* and *Proactive Ant Sampling* [41].

In the Pheromone Diffusion process, the pheromone information placed by RBAs is broadcasted periodically to neighbor nodes so that it becomes possible for them to calculate their pheromone information. Furthermore, neighbor nodes also broadcast their pheromone information to their neighbors. This Pheromone information spread via PD messages is considered a virtual pheromone, and it does not override the regular *pheromone* placed by Backward Ants. This pheromone information is used for *proactive forward ants*.

In the Proactive Ant Sampling process, the source node of the communication session periodically sends Proactive Forward Ants (PFWAs) towards destination nodes. PFWA messages are forwarded by each node in a stochastic way, i.e., calculating the next node probabilistically using the virtual and regular pheromone information. Just like RFWAs, once PFWAs reached the destination node, they are converted back into PBAs and retrace their path back to the source node while updating the pheromone information on each node on their way back.

With these processes, pheromone tables are created and updated at each node. When a data packet is sent from the source node, the next hop of the data packets is selected by another probabilistic formula that uses only regular pheromone information. It should be noted that while PFWAs are forwarded in a more exploratory way, data packets are forwarded more securely.

The AntHocNet algorithm also contains a mechanism to detect link failures and tries to repair them locally. When a link fails, the node detecting the link failure updates its pheromone tables and notifies its neighbors about this event. It is also possible that the node detecting the link failure initiates a local route repair process in which it sends Forward Ants similar to RFWAs, but they have limited lives. This limitation is placed so that it does not travel far from the failed link. These forward ants reach the destination and are converted into backward ants and return to the node that detected the link failure by updating the pheromone tables on their way back to the node.

To sum up, AntHocNet is a bio-inspired routing algorithm that contains reactive and proactive components. It iteratively tries to improve the established route. This improvement effort also brings adaptability to the changes that occurred within the network topology.

2.3 Mobility Models In Ad Hoc Networks

Mobility Models concern how the nodes move in an ad hoc network. It is a way to express how the nodes' positions, velocities, and accelerations change in time [12]. Determining node mobility is a significant part of designing an ad hoc network application because it directly affects the performance of routing algorithms and should not be overlooked [42], [43], [44], [45].

In the literature, mobility models are divided into several categories. Bujari et al. divide the mobility models for FANETs into five different categories, namely *randomized*, *time/space dependent*, *path planned*, *group* and *topology control based* [46]. On the other hand, Guillen-Perez and Cano use the first four categories; however, they offered *hybrid* category instead of *topology control based* category as the fifth one [12]. Although these mobility model categories are mainly proposed for FANET applications, some have been used by MANETs

and VANETs for decades. For instance, randomized models like Random Way Point and Random Walk mobility models are used extensively for MANETs.

For VANETs, the importance of realistic mobility models increases, raising the mobility model's complexity. According to Harri et al., a realistic vehicular mobility model should consider the traffic, vehicle characteristics, street topologies, obstacles, human driving patterns, attraction, and repulsion points [44]. On the other hand, in the Manhattan Grid mobility model, nodes randomly move either horizontally or vertically by assuming a grid road topology [45]. Apart from the above-mentioned realistic mobility models, mobility models like the Manhattan Grid are also used for VANETs [12].

The randomized mobility models are the most common mobility models for ad hoc networks. They are the simplest models where the speed, direction, acceleration of each node change independently and uniformly distributed randomly. Moreover, in randomized mobility models, nodes do not take into account their past movements; in other words, no feedback mechanism affects the decision of nodes about how to move [46]. The most common randomized models are Random Walk, Random Waypoint, Random Direction, and Manhattan Grid [12].

Time-dependent models are mobility models which use a formula to avoid sudden changes in a node's direction and speed [46]. The main idea behind this group is that the smooth changes in node velocity vector are more realistic and better fit for many ad hoc network applications. The prominent representatives of this group are the Gauss-Markov mobility model, Enhanced Gauss-Markov mobility model, and Smooth Turn mobility model. In Gauss-Markov and Enhanced Gauss-Markov mobility models, the smooth changes in direction and speed are provided by using the previous state, i.e., speed and direction, of the node as an input for the equation through which new speed and direction are calculated [47]. In the Smooth Turn mobility model, sudden changes in the node's direction and speed are avoided by making the node move in a circular path. The center of the circular path is chosen as perpendicular to the node's current direction, which provides a smooth turning path for the node [12].

Path-planned models use a set of predefined patterns and make nodes use a pattern from the provided set. The selection of a pattern from the set occurs randomly. However, node movements fit one of the predefined patterns without any randomization. The most prominent

representatives of this group are the Semi-Random Circular Movement and Paparazzi Model [12]. These models are FANET specific models, and they are created by modeling UAV movements. Paparazzi Mobility Model, for instance, is a mobility model that offers a set of UAV movements that are taken from the open-source Paparazzi Free Autopilot project [48]. It is a realistic mobility model for FW-UAVs [49].

Lastly, another important category of mobility models that should be mentioned is the group mobility models. The models in this category use the idea that a subset of the nodes moves together within a swarm of nodes. The group mobility patterns specially fit some of the FANET applications [12]. The most known representatives of this category are Reference Point Group Mobility (RPGM), Column (CLMN), Nomadic Community (NC), and Particle Swarm Mobility Model (PSMM).

CHAPTER 3

RELATED WORK

The routing issue in multi-UAV applications and systems is a significant problem and worked by many researchers, especially in the last decade. Different from nodes of MANETs and VANETs, the unique characteristics of UAVs, such as high mobility, non-uniform distribution over a 3D space, and mobility regime, make routing in FANET applications a complex issue [50]. In order to provide reliable communication between UAVs, many routing algorithms are proposed to address the FANET application needs and UAV characteristics. Apart from network architecture and routing protocols, mobility models and simulators/testbeds are also studied widely.

In [51], UAVs are mainly categorized as Fixed-Wing UAVs and Rotary-Wing UAVs. The similarities and differences between FW-UAVs and RW-UAVs, such as energy constraints, speeds, and mobility, are discussed. Moreover, FANET communication architectures, such as single-layer & single-group, single-layer & multi-group, multi-layer & multi-group, are investigated and offered to various FANET application areas. This discussion includes the communication technologies suitable for UAV-To-UAV communications and UAV-To-Infrastructure communications such as IEEE 802.11n/b/g/a, IEEE 802.15.4, and IEEE 802.14.4, LTE advanced, LTE, CDMA, TDMA, GPRS, and GSM [52]. Moreover, a FANET routing protocol taxonomy is given with the following main categories: Topology-based routing, Geographic routing, Hybrid of Topology-based and Geographic routing, Bio-inspired routing. In this taxonomy, variations of OLSR and AODV find themselves a place while AntHocNet is not given under the Bio-inspired routing algorithms.

Oubbati et al. provide an altitude point of view on UAV classification based on the altitude of UAVs, which includes High Altitude UAVs, such as satellites and airships, Medium Altitude UAVs such as aircraft, and Low Altitude UAVs such as drones and copters [50]. Due to this wider perspective, the study in [50] thoroughly discusses UAV-To-UAV (U2U) communications, UAV-To-GROUND (U2G) communications and UAV-To-SATELLITE (SATCOM) communications and compares them in terms of *line of sight (LOS)*, *cost*, *coverage* and *exploitation*. This study also provides a protocol taxonomy that includes Topology-based, Secure-based, Swarm-based, Hierarchical-based, Energy-based, Heterogeneous-based, Position-based, and DTN-based categories for routing algorithms. Although many routing algorithms, including AODV and OLSR, are referenced within this taxonomy, AntHocNet is not listed under the category of swarm-based routing protocols.

In [1], UAV classification is presented according to both altitude and type. According to altitude, UAVs are divided into two categories: High Altitude Platform (HAP) and Low Altitude Platform (LAP). According to type, UAVs are categorized into Fixed-wing and Rotary-wing UAVs. Moreover, this study mainly focuses on two use cases: Aerial Base Stations and Cellular-Connected Users. For both of these cases, the key challenges, solutions, and open problems are presented.

In [53], it is claimed that the routing protocols proposed for MANET applications do not fit well to FANET applications due to that FANET applications have different requirements than MANET or VANET applications, such as mobility patterns and power constraints. It is stated that OLSR is not a suitable protocol for FANET applications due to the high routing overheads, and AODV is not suitable due to the higher end-to-end delays. However, several studies evaluate the performance of AODV and OLSR routing algorithms in FANET Applications. In [31], a performance comparison of AODV, DSDV, and OLSR routing protocols for FANETs is given. The evaluation uses the NS-2 network simulator. Also, packet delivery ratio, end-to-end delay, and throughput are the selected metrics. In this study, it is claimed that OLSR performs better than AODV and DSDV protocols in FANETs. A similar research is conducted in [28] to compare the performance of AODV, DSDV, DSR, OLSR, AOMDV, HWMP routing protocols for FANETs using an NS-2 simulation environment. The performance of OLSR and HWMP routing protocols are claimed to perform better than AODV,

DSDV, DSR, and AOMDV routing protocols concerning packet delivery ratio, end-to-end delay, and throughput. Although these studies are done for FANET applications, both of them conduct simulations in 2D environments. In [54], a similar study is conducted in a 3D environment. Maakar et al. compared OLSR and AODV routing protocols in a 3D FANET environment using an NS-3 simulator [54]. The mobility model used in this study is the Gauss-Markov mobility model, and it is claimed that OLSR performs better than AODV according to the simulation results. In [55], evaluation of AODV, OLSR, and DSDV routing protocols for RW-UAVs is presented. The simulations are done in a 3D space using Gauss-Markov, and Random Waypoint mobility models with node speeds, 5m/sec and 10m/sec. In [56], AODV and OLSR routing protocols are investigated in the NS-2 simulation environment. This research presents the packet delivery ratio provided by AODV and OLSR routing protocols using node speed, transmission range, and the number of nodes as independent parameters. The simulations are conducted for a 2D area and node speeds in the range of 0 - 60 m/sec.

In [39] and [57], swarm-based routing algorithms, namely AntHocNet and BeeAdHoc, are offered for FANETs. It is claimed that AntHocNet and BeeAdHoc algorithms can be efficiently used for FANET applications by providing the network performance comparison of AntHocNet, BeeAdHoc, AODV, DSDV, and DSR routing protocols. The simulation is conducted for a 2D area using the Random Waypoint mobility model with node velocities between 20 m/sec and 50 m/sec. The performance metrics used in this study are throughput, end-to-end delay, and routing overhead. In [38], AntHocNet is offered for FANETs also, and the network performance of the AntHocNet routing algorithm is compared with AODV, DSR, and DSDV routing algorithms in terms of throughput, end-to-end delay, and routing overhead. The simulations are conducted within a 2D environment and with an RWP mobility model with node velocities between 5 m/sec and 50 m/sec. Although AntHocNet routing algorithm is studied and proposed by [39], [57] and [38] for FANETs, the mobility models used in these researches, namely RWP, are not appropriate for FW-UAV Networks, and simulations are conducted in 2D environments with relatively lower node speeds.

In literature, almost all of the routing algorithm performance evaluations are done by using simulators. This is mainly because building the necessary topology with the required nodes,

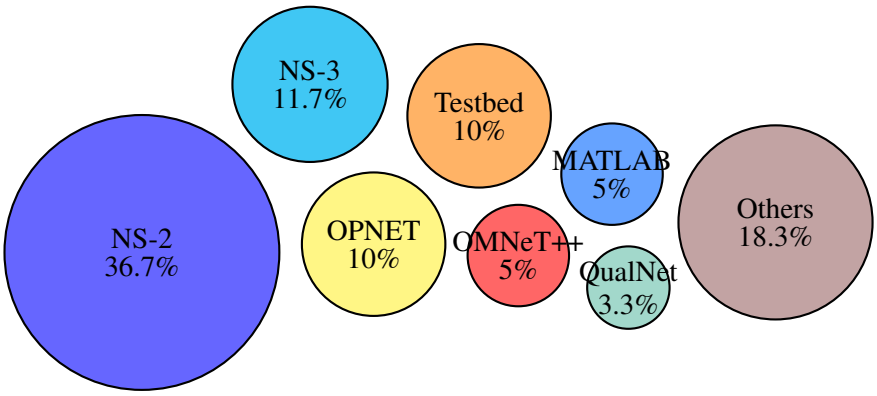


Figure 2: Simulators used in FANET routing protocols [50]

such as drones, aircraft, or copters, is costly; and it is hard to repeat the tested scenario [49]. Oubbati et al. provide comparative research of 60 studies concerning FANET routing protocols where only the %10 of the studies used testbeds as shown in Figure 2 [50]. Simulators, such as NS-2, NS-3, OPNET, OMNeT++, QualNet, and MATLAB, are preferred in most studies. Since the simulators are used to evaluate the performance of the routing protocols, the issue of modeling the mobility of nodes becomes crucial and demanded [51].

Many studies point out that the simulation results change meaningfully according to which mobility model is used. In one of the early studies concerning the issue of the routing algorithm performance under different mobility models, it has been shown that the performances of Ad-Hoc Wireless Network routing protocols, namely AODV, DSDV, and DSR, differ under different mobility models, namely RWP, Column, Pursue and RPGM-RW [58]. There is a meaningful difference in the routing protocol performances for the selected QoS metrics, such as packet delivery ratio and packet routing overhead. It is stated that some routing algorithms perform better in some mobility models; for instance, DSR shows better performance in Pursue and RWP mobility models while AODV shows better performance with Column and RPGM-RW mobility models.

In [59], the performances of AODV, OLSR, GRP, RGR, and flooding protocols are compared. It is shown that the selected QoS metrics, namely control overhead, packet delivery ratio, and average delay, differ significantly according to which of the RWP, EGM, ST, RD, or GM mobility model is selected. It is concluded that the RWP mobility model is not suitable for Unmanned Arial Ad-hoc Networks, in other words, FANETs. At the same time, it is

appropriate to use the EGM mobility model because the EGM mobility model provides more realistic trajectories than the RWP mobility model. This study also states that OLSR and GRP protocols are more responsive to the change of network topology than AODV.

In [42], AODV and AOMDV routing protocol performances are compared under different mobility models, namely Manhattan Grid, RPGM, Column, RWP, Pursue, Nomadic. It is shown that there is a meaningful difference in network performance metrics, namely packet delivery ratio, throughput, and end-to-end delay, under these different mobility models. In [60], performance evaluations of AODV, DSR, and OLSR routing protocols under different mobility models, namely RPGM, GM, RWP, and Manhattan Grid, are presented. Simulations have been conducted with NS-2.35 with node speeds 10, 50, 100, 200, 300, 400, 500 m/sec. The compared metrics are packet delivery ratio, end-to-end delay, and routing overhead. Also, in this research, a significant effect of the mobility model on network performance can be spotted.

In [61], AODV, DSDV, and OLSR routing protocol performances are investigated with different mobility models, namely RPGM, RWP, and Pursue. It is presented that all of the selected routing protocols provide different packet delivery ratios under different mobility models. In [62], OLSR and G-OLSR routing protocols are used in 3D FANET simulations in order to investigate the performance differences of these routing protocols under different mobility models, namely Gauss Markov and RWP.

Because the mobility model affects the simulation results significantly, selecting the appropriate mobility model for the research becomes essential. In [47], it is stated that the widely used RWP mobility model is not a suitable model for UAVs because sudden stops and sharp turns reported in the RWP mobility model do not match the trajectories of UAVs. For this reason, the authors propose Enhanced Gauss-Markov mobility model in 2D, which ensures that the nodes move neither with sharp turns nor with sudden stops. It is stated that different from ST and GM, EGM also provides smooth turns on the boundaries of the simulation area, which is a better fit for FW-UAVs. In [63], the 2D EGM mobility model proposed in [47] is extended to 3D. This study compares the 3D EGM mobility model with the GM mobility model with AODV routing protocol by using NS3. In another study [64], OLSR, AODV, DSR, TORA,

and GRP routing protocols are compared in different mobility models, namely PRS, MGM, RWPM, and SCRM.

In [49], the Paparazzi mobility model (PPRZM) for UAV ad hoc networks is proposed. In this research, it is emphasized that in order to address the network problems adequately, the use of realistic mobility models is necessary. It is shown that PPRZM provides trajectories for nodes that are akin to the actual UAV trajectories. Moreover, since the node trajectories provided by the PPRZM mobility model are geometrically more similar to real traces than the trajectories provided by the RWP mobility model, the QoS metrics measured for the PPRZM are more similar to the metrics measured by Real Traces than that of RWP.

In the light of these studies, it can be seen that the literature has a gap concerning the use of bio-inspired routing algorithms in FW-UAV ad hoc networks. Even though there are some researches, such as [38], [39], [57], which study the AntHocNet routing algorithm for FANETs, these studies do not take the effect of using different mobility models into consideration and conducted their simulations in a 2D environment. However, there are several studies, such as [58], [59], [47], [49], that point out the fact that the mobility model selection has a meaningful impact on the measured network-related metrics. Moreover, most of the works that focus on routing protocol in FANETs do not provide 3D simulation results. However, it is necessary to consider the 3D nature of FANET applications while working in FANET routing algorithms.

Another point that should be mentioned is that the literature lacks studies dealing with the effect of topology change on the lower threshold of the average number of neighbors required for connectivity. In [7], the minimum threshold is presented for the average number of neighbors in WSNs. However, in this study, nodes are stationary since the focus of the study is WSNs.

When the abovementioned points are considered, this thesis fills these gaps by executing simulations of various scenarios. In the preliminary simulations, the effect of the topology change on the minimum value of the average number of neighbors to provide network-wide connectivity is presented. Then, simulations are conducted to compare the performances of the traditional routing algorithms, namely AODV and OLSR, and the bio-inspired routing algorithm, namely AntHocNet. Since FANETs and FW-UAVs are the focal points of this

research, simulations are executed in a 3D simulation space. Moreover, two mobility models, EGM and PPRZM, are selected aside from RWP; because these mobility models are offered for FW-UAVs and represent different categories of the mobility models.

CHAPTER 4

METHODOLOGY

In this chapter, the methodology of this study is presented. This experimental research compares the network performances of bio-inspired routing algorithms, specifically AntHocNet, and traditional routing algorithms, namely OLSR and AODV in FANET applications consisting of FW-UAVs. Selected metrics for comparing network performance are mean end-to-end delay, mean end-to-end delay variation (a.k.a., jitter), packet loss ratio, mean hop count (i.e., the mean path length), and routing overhead.

4.1 Identification of Variables

The variables, i.e., parameters, used in this research can be divided into three main categories: dependent variables, independent variables, and constants.

Dependent variables are the metrics intended to be measured to provide a comparison of network performances. These variables are the performance indicator metrics of a routing algorithm, namely end-to-end delay, end-to-end jitter, packet loss ratio, routing overhead, and mean hop count.

Four independent variables should be mentioned. Three of these independent variables are the mobility model, node speed, and transmission data rate, which affect the environment and simulation scenario by changing the topology change rate and network load. Besides, the last one is the routing algorithm which is one of the main focusing points of this research.

Constants are the other necessary parameters that need to be defined in order to execute the simulation scenarios. These variables are the simulation space, the simulation time, the propagation delay model, the propagation loss model, the transport protocol, the transmission range, the number of nodes, the data packet size, and the communication standard.

There is also a vital constant, *average number of neighbors*, which is a function of other constants. The *average number of neighbors* is calculated for a 3D simulation space as follows:

$$AvgNumOfNeighbors = \left(NodeNum * \left(4/3 * \pi * TxRange^3\right)\right) / SimSpace \quad (1)$$

, where *TxRange* is the transmission range in meters, *NodeNum* is the number of nodes, *AvgNumOfNeighbors* is the average number of neighbors, and *SimSpace* is the volume of simulation space in cubic meters.

The number of nodes that can be reached by one hop gives the average number of neighbors. In other words, the nodes that are within the transmission range are considered neighbor nodes. In Equation 1, the ratio of the number of nodes over simulation space indicates the node density. The multiplication of node density with the volume of a sphere with a radius of *TxRange* gives us the average number of neighbors, i.e., the number of nodes that fall into the volume of a sphere with a radius of transmission range.

4.2 Research Design

In this section, the components of the design of this research are provided. The given components are the methodology of this research, simulation components, data collection process, and simulation scenarios.

4.2.1 Research Methodology

In this experimental research, in order to answer the research questions raised in Section 1, NS-3 simulator [65] is used. NS-3 is the newer version of the NS-2 simulator, which

researchers widely use to execute network simulations. The main difference of NS-3 is that it is entirely C++ based, and it can support Python scripts. Since NS-3 is provided with source codes, it can be extended and used according to the simulation needs.

By using the NS-3 simulator, the network performances of different routing algorithms are studied. Two groups of algorithms are used, namely the control group and the experimental group. As for the control group, AODV and OLSR routing algorithms are selected, and the implementations provided by the NS-3 simulator are used. In other words, a reactive and a proactive routing protocol are selected for the control group. As for the experimental group, AntHocNet, a bio-inspired routing algorithm, is implemented.

For both of these groups, 20 simulations are executed for each simulation detailed in Section 4.2.4. These simulations use the dependent parameters of end-to-end delay, end-to-end delay variation, packet loss ratio, and routing overhead. As independent parameters, node speeds, data rate, and mobility models are used. The remaining parameters used in simulations are given in Table 1.

In [7], it is stated that to ensure network-wide connectivity in a randomly deployed wireless network with stationary nodes, the average number of neighbors for each node should exceed 5.5. Consequently, if the number of neighbors is small, the network nodes may not be able to reach each other due to a partitioned network. On the other hand, if the number of neighbors is large (i.e., the network is dense), the network suffers from high routing overhead problem. Consequently, in order to ensure that the operating region is neither in the no-connectivity region nor the high routing overhead region, we set the average number of neighbors is set to 7.54 by deploying 50 nodes with transmission ranges of 600 meters in a 2000 m x 2000 m x 1500m simulation space.

In the following subsection, the results of the preliminary simulations are presented to clarify why the value of 7.54 is appropriate as the average number of neighbors. In this verification, the effect of the average number of neighbors on connectivity is provided by choosing the average number of neighbors as an independent parameter and the packet loss ratio as a dependent parameter. Also, the effect of node mobility on the minimum value of the average number of neighbors required for connectivity is inspected.

Table 1: Parameters Used in Simulations

Constant Simulation Parameters	
Simulation Time	120s
Communication Standard	IEEE 802.11g
Propagation Delay	ConstantSpeedPropagationDelayModel
Propagation Loss	RangePropagationLossModel
Simulation Space	2000m x 2000m x 1500m
Transmission Range	600m
Number Of Nodes	50
Average Number Of Neighbors	7.54
Transport Protocol	UDP
Network Traffic	Constant Bit Rate
Data Packet Size	256 bytes
Other Simulation Parameters	
Simulator	NS-3.26
Num of Runs	20
Mobility Models	Paparazzi Mobility Model (PPRZM) Enhanced Gauss-Markov (EGM) Random Waypoint (RWP)

4.2.1.1 Average Number of Neighbors Required to Ensure Connectivity

In order to verify the value chosen for the average number of neighbors, preliminary simulations are conducted. In these simulations, two different scenarios are executed.

Preliminary Simulation 1 is executed for the stationary case. In other words, node speed is selected as 0 m/s. In this scenario, the number of nodes is set to 30, and the transmission range spans from 300 m to 750 m. As a result, the average number of neighbors spans from 0.57 to 8.84. For these metrics, the packet loss ratios for AODV, OLSR, and AntHocNet algorithms are given in Figure 3.

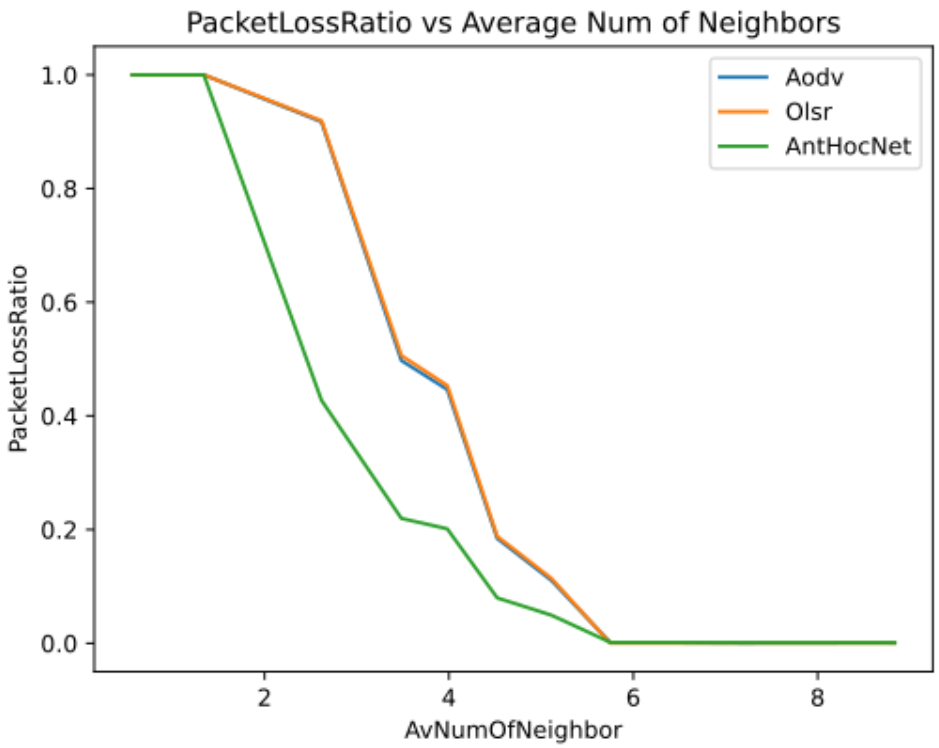


Figure 3: Packet Loss Ratio vs Average Number Of Neighbors, Node Speed = 0 m/s

It should be noted that when the transmission range is above 650 m, the packet loss ratio is below 0.1 for all three of the routing protocols. On the other hand, when the transmission range is going to 300 m, the packet loss ratio goes up to 1. This region, the region where the packet loss ratio is higher than 0.9, is called the no-connectivity region; because, as it is shown

in [7], the network is likely to be partitioned due to the insufficient number of neighbors. It should be noted that when the transmission range is 650 m, the value of the average number of neighbors becomes 5.75; and when the transmission range is 625 m, the value of the average number of neighbors becomes 5.11. This situation can be observed from Figure 3, where the packet loss ratio of AODV, OLSR, and AntHocNet algorithms is given according to the average number of neighbors.

The results of the preliminary simulation 1 are compatible with the results provided in [7]. In other words, when the node speed is 0 m/s, and the value of the average number of neighbors becomes lower than the threshold value stated in [7], i.e., 5.5. In this case, the network is potentially partitioned, and consequently, the packet loss ratio increases dramatically. On the other hand, when the average number of neighbors exceeds 5.5, a non-partitioned network can be established.

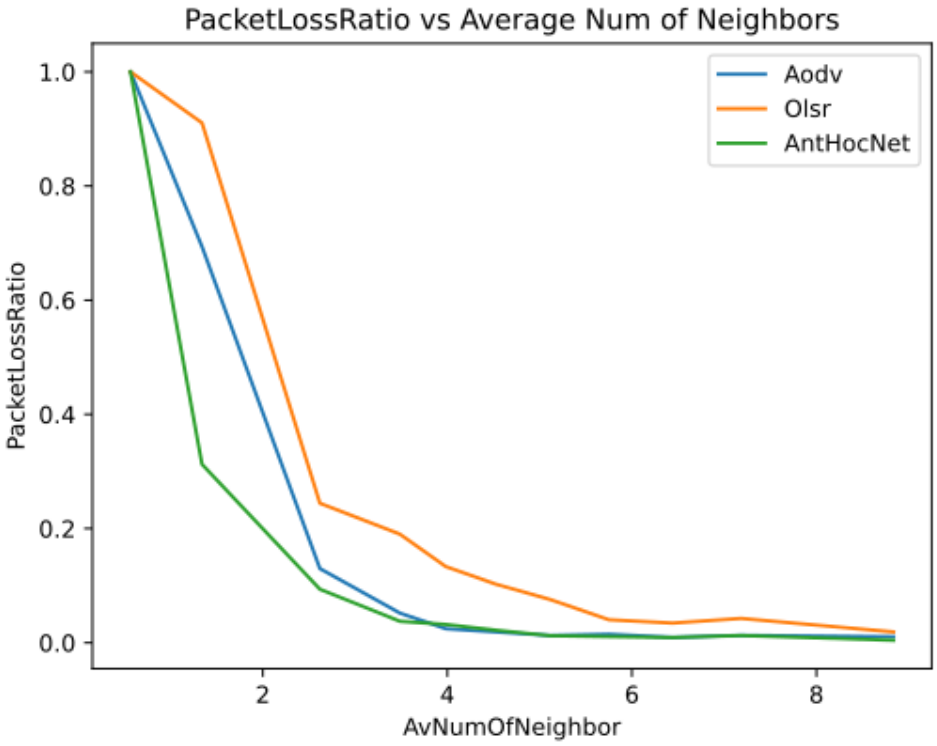


Figure 4: Packet Loss Ratio vs Average Number Of Neighbors, Node Speed = 5 m/s

Preliminary Simulation 2 is executed for the case where the nodes are mobile. In this case, simulations are executed with node speeds 5m/s with the RWP mobility model. Apart from the mobility of nodes, other parameters are the same with preliminary scenario 1. The packet loss ratio obtained for preliminary simulation two is presented in Figure 4.

From the Figures 3 and 4, it can be observed that as node speed increases from 0 m/s to 5 m/s, the routing protocols can establish paths through the network at lower transmission range values, i.e., at the lower average number of neighbor values, at the extend of their abilities. The values of the transmission range (TxR), the average number of neighbors (ANoN), and the packet loss ratio (PLR) are provided for two points, namely P1 and P2, in Table 2. These two points indicate the region where the PLR value becomes smaller than 0.1; in other words, the network-wide connectivity is established at some point between P1 and P2.

Table 2: Node Speed vs Transmission Range Lower Threshold Data Points

Effect of Node Speed On the Average Number of Neighbors (ANoN)							
Protocol	Speed	TxR_{P1}	ANoN_{P1}	PLR_{P1}	TxR_{P2}	PLR_{P2}	ANoN_{P2}
AODV	0	625	5.11	0.229	650	5.75	0.00022
AODV	5	500	2.61	0.179	550	3.48	0.034
OLSR	0	625	5.11	0.23	650	5.75	0
OLSR	5	600	4.52	0.12	625	5.11	0.08
AntHocNet	0	625	5.11	0.101	650	5.75	0.0014
AntHocNet	5	500	2.61	0.11	550	3.48	0.049

It is clear from the figures and the table that as node speeds increase from 0m/s to 5m/s, the lower threshold value for the average number of neighbors decreases because of the decrease in transmission range where the packet loss ratio drops below 0.1. These results clearly show the effect of node speed on the required lower threshold of the average number of neighbors for the non-partitioned network.

Simulations with different node speeds are conducted for the AntHocNet routing protocol to clarify the effect of node speed further. In Figure 5, the packet loss ratio for the AntHoc-

Net routing protocol is given according to the average number of neighbor values with four different node speeds.

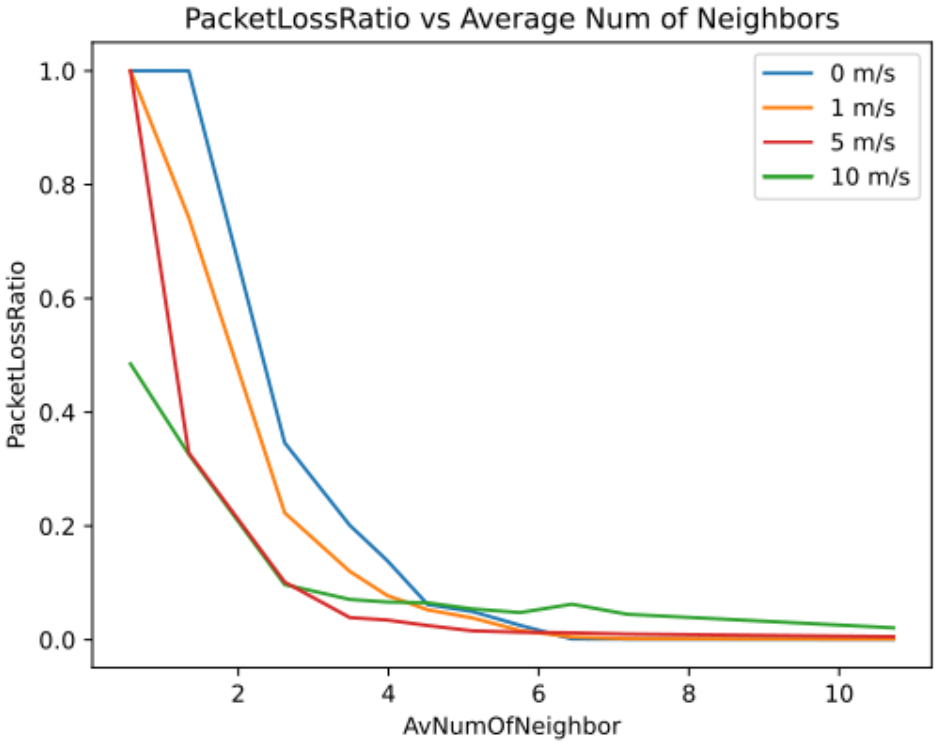


Figure 5: Packet Loss Ratio vs Average Number Of Neighbors, AntHocNet

As presented in Figure 5, an increase in node speed lowers the required average number of neighbors for a non-partitioned network. As node speed increases, it is harder to reduce the packet loss ratio to 0, even when the value of the average number of neighbors is higher than 5.5. Because as node speed increases, the network topology changes faster, and it gets more challenging for the routing protocol to deliver packets to their destinations.

The preliminary simulation results show that the selected value for the average number of neighbors, i.e., 7.54, provides a meaningful operating point for practical FW-FANET applications where nodes are mobile.

The raw data, obtained from these preliminary simulations, is provided in Appendix A.

4.2.2 Simulation Components

This study aims to deal with FANET applications consisting of FW-UAVs. Due to the nature of FANETs, it is necessary to provide some simulation components unique to FANET applications. For this reason, firstly, the necessary components to simulate FW-UAVs are presented. Also, the data collection mechanism is explained. Then, the simulation scenarios appropriate to FW-FANET applications are defined to answer the stated research questions. After that, by using the simulation components and NS-3 simulator, defined scenarios are executed to collect data. Finally, the collected data is analyzed to be able to answer the research questions.

In this section, the components of simulations are given in more detail. This section aims to explain the simulation environment and provide the details of some of the components.

Simulations conducted in this research are composed of five main components: nodes, network stack, node mobilities, routing algorithm, and network application.

4.2.2.1 Nodes

Nodes are the basic building blocks of an NS-3 simulation. They are the entities that run the network stack, routing algorithm, and network application. In the NS-3 simulator, the mobility models provide the next position for nodes, routing algorithms are executed by each node, and network applications are installed to the nodes. In Figure 6, a sample network is depicted with the NS-3 visualizer tool where the nodes are shown as red discs.

4.2.2.2 Network Stack

NS-3 simulator tool provides various network stack modules to be used in simulations. For the PHY layer of the network stack, IEEE 802.11g standard is used. For the MAC layer, *AdHocWiFiMac* provided by NS3 is used. As for communication channel management, Yans Wifi Channel, which is an implementation of *Yet Another Network Simulator* [66] widely used in wireless network simulations, is used with *RangePropagationLossModel* and *Con-*

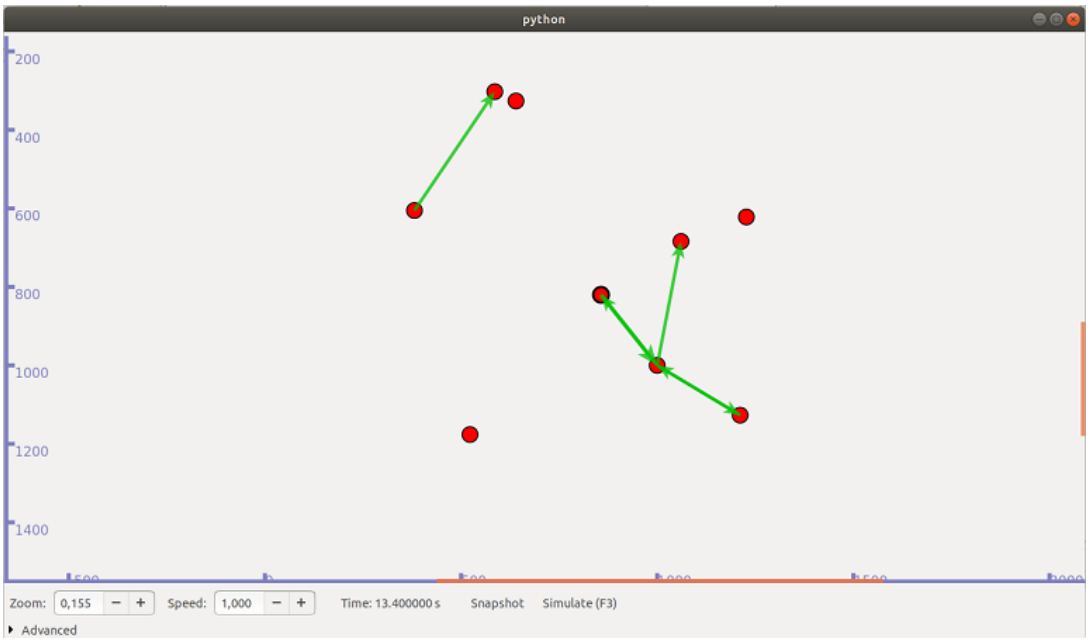


Figure 6: A screenshot of NS-3 Visualizer

stantSpeedPropagationDelayModel as propagation delay and propagation loss models. In Figure 7, the selected modules and properties for the network stack are shown as NS-3 code blocks.

4.2.2.3 Node Mobilities

In NS-3, node mobilities are governed by mobility models. The installed mobility model provides the next position for each node, and nodes move to these positions in their successive iterations. In this research, three mobility models are selected from different mobility model categories. First of the selected mobility models is the EGM mobility model, which is an extension of the Gauss-Markov(GM) mobility model [46]. The EGM mobility model is a time-dependent mobility model that uses the previous mobility state information to avoid sharp movements [47]. The second selected mobility model is the PPRZM mobility model, which falls under the category of path-planned mobility models [12]. The last selected mobility model is the RWP, one of the most used randomized mobility models [46].

```
WifiMacHelper wifiMac;
wifiMac.SetType ("ns3::AdhocWifiMac");

YansWifiPhyHelper wifiPhy = YansWifiPhyHelper::Default();

YansWifiChannelHelper wifiChannel;
wifiChannel.SetPropagationDelay("ns3::ConstantSpeedPropagationDelayModel");
wifiChannel.AddPropagationLoss("ns3::RangePropagationLossModel",
 ↪  "MaxRange", DoubleValue(p_config->transmissionRange));
wifiPhy.SetChannel(wifiChannel.Create ());

WifiHelper wifi;
wifi.SetStandard(WIFI_PHY_STANDARD_80211g);
wifi.SetRemoteStationManager("ns3::ArfWifiManager");

devices = wifi.Install (wifiPhy, wifiMac, nodes);
```

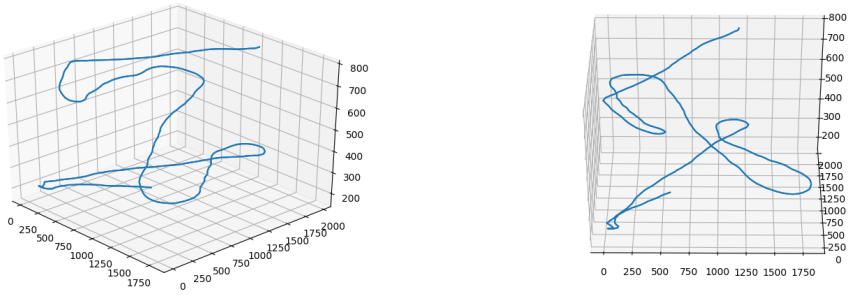
Figure 7: Network Stack User Code

In NS-3, the RWP mobility model is available and can be used in 3D space. However, NS-3 does not provide implementations for EGM and PPRZM. Therefore, 3D implementations for these mobility models are developed within the scope of this research.

EGM Mobility Model is implemented according to the [47]. Different from the GM mobility model, the EGM mobility model implements a boundary avoidance algorithm. Thanks to the boundary avoidance mechanism, nodes can avoid boundaries and smoothly turn back into the designated space. To execute this mechanism, the EGM model defines not only outer bounds but also inner bounds. When a node passes over inner boundaries, the boundary avoidance algorithm changes the mean direction or mean pitch value which affects the direction or pitch of the node according to the value of alpha. The generic formulation in calculating new velocity, direction, and pitch is given in Equation 2, where *rv* is a random number drawn from the Gaussian distribution with zero mean and unit variance.

$$next\_val = \alpha * current\_val + (1 - \alpha) * mean\_value + \sqrt{(1 - \alpha^2)} * rv \qquad (2)$$

As seen from the Equation 2, changing the mean value of direction or pitch affects the value used in the next iteration according to the value of α, which is the memory constant of this model. α can take values between 0 to 1. In [47], it is taken as 0.86. In simulations conducted for this research, the α value is taken as 0.85, as can be seen from the EGM User Code given in Appendix B.



(a) EGM Trajectory-Perspective 1 (b) EGM Trajectory-Perspective 2

Figure 8: An Example of EGM Trajectories

Implemented EGM mobility model uses Equation 2 in order to calculate the next velocity, direction, and pitch values. Velocity, direction, and pitch values are used to define a 3D velocity vector, where the value of direction is between 0 and 2π, the value of pitch is between $-\pi/2$ and $\pi/2$ and the value of velocity is a positive real number. However, since NS-3 uses velocity vectors on the cartesian XYZ frame, the coordinate frame is converted after calculating the next velocity, direction, and pitch values. In Figure 9, the code block that is used to calculate the next velocity vector is presented.

A sample trajectory generated by the EGM mobility model is given in Figure 8. Figures 8a and 8b provide the sample trajectory from different perspectives.

PPRZM Mobility Model is implemented according to [49]. PPRZM is a stochastic model where nodes randomly select one of the pre-determined trajectory patterns shown in Figure

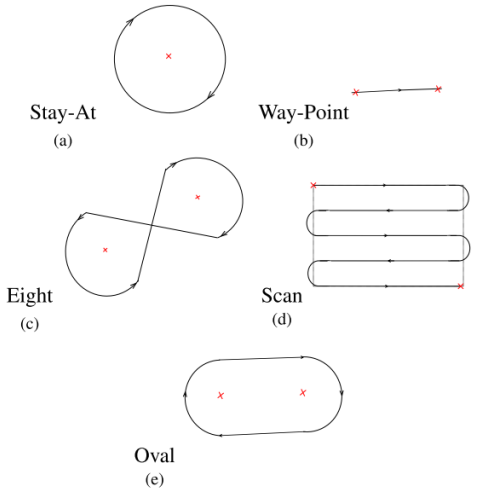
```
1  double rv = m_normalVelocity->GetValue ();
2  double rd = m_normalDirection->GetValue ();
3  double rp = m_normalPitch->GetValue ();
4
5  //Calculate NEW velocity, direction, and pitch values using Gauss-Markov
6  double one_minus_alpha = 1 - m_alpha;
7  double sqrt_alpha = std::sqrt (1 - m_alpha*m_alpha);
8  m_Velocity = m_alpha * m_Velocity + one_minus_alpha * m_meanVelocity +
   ↪  sqrt_alpha * rv;
9  m_Direction = m_alpha * m_Direction + one_minus_alpha * m_meanDirection
   ↪  + sqrt_alpha * rd;
10 m_Pitch = m_alpha * m_Pitch + one_minus_alpha * m_meanPitch + sqrt_alpha
   ↪  * rp;
11
12 //Calculate linear velocity vector to use with constantVelocityHelper
13 double cosDir = std::cos (m_Direction);
14 double cosPit = std::cos (m_Pitch);
15 double sinDir = std::sin (m_Direction);
16 double sinPit = std::sin (m_Pitch);
17 double vx = m_Velocity * cosDir * cosPit;
18 double vy = m_Velocity * sinDir * cosPit;
19 double vz = m_Velocity * sinPit;
20 m_helper.SetVelocity (Vector (vx, vy, vz));
```

Figure 9: EGM Velocity Calculation

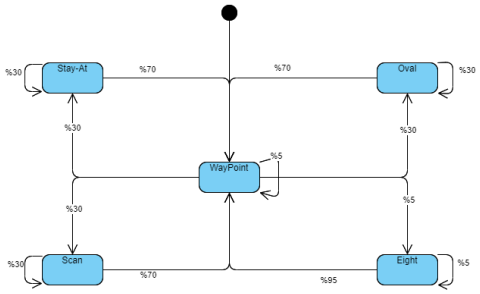
10a. Each node starts with the *Way-Point* state in which the destination point is chosen randomly. Then, the nodes probabilistically change their states according to the state diagram shown in Figure 10b. Note that a node can change its movement pattern only through moving to another random point within the boundaries, i.e., only through passing *Way-Point* state.

In [49], the probability of occurrence of each movement pattern is given as follows:

- Probabilities of *Stay-At*, *Oval* and *Scan* : %30
- Probabilities of *Eight* and *Way-Point* : %5



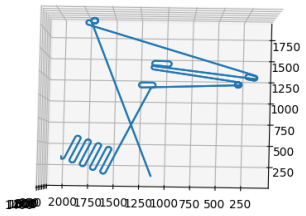
(a) PPRZM Movements (Bouachir, 2014)



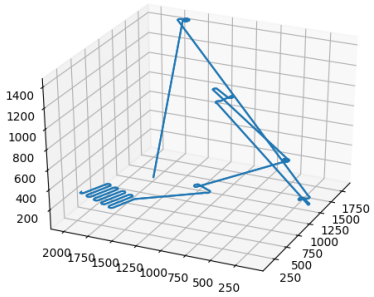
(b) PPRZM State Diagram (Bouachir, 2014)

Figure 10: PPRZM States

According to these probabilities, it can be claimed that the PPRZM mobility model causes a lower topology change rate than the EGM mobility model. Because, when a node is in *Stay-At* or *Oval* state, the displacement of the node decreases.



(a) PPRZM Trajectory-Perspective 1

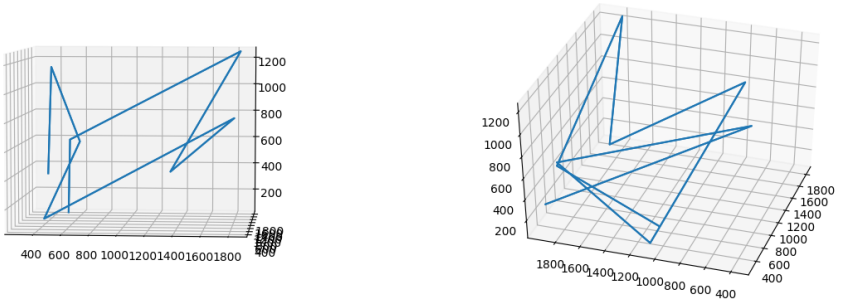


(b) PPRZM Trajectory-Perspective 2

Figure 11: An Example of PPRZM Trajectories

A sample trajectory generated by the PPRZM mobility model is given in Figure 11. The sample trajectory is given from different perspectives in Figures 11a and 11b. Also, NS-3 implementation of PPRZM is given in Appendix B.

RWP Mobility Model is one of the fundamental mobility models which is used widely within the literature. Even though many studies do not recommend the RWP mobility model for FANETs, this research uses the RWP mobility model to provide a common base with other studies. NS-3 simulator provides a 3D implementation of the RWP mobility model, which is used in this study. In Figure 12, a sample trajectory of a node generated by the RWP mobility model is given, and Figures 12a and 12b display the sample trajectory from different perspectives.



(a) RWP Trajectory-Angle 1

(b) RWP Trajectory-Angle 2

Figure 12: An Example of RWP Trajectories

4.2.2.4 Routing Algorithms

In NS-3 simulations, routing algorithms are part of the internet stack (Figure 13) which implements Layer 3 (L3) and Layer 4 (L4) of the network stack, i.e., Network Layer and Transport Layer. The internet stack is installed on the nodes to use L3 and L4 in their network communications.

In this research, the selected routing algorithms used and compared in simulations are OLSR, AODV, and AntHocNet. In the following, these algorithms are briefly explained.

OLSR routing algorithm is a traditional proactive routing algorithm whose fundamental mechanism is explained in Section 2.2.1. As stated, the NS-3 implementation of the OLSR

```
InternetStackHelper stack;
//has effect on the next Install ()
stack.SetRoutingHelper (*pIpv4RoutingHelper);
stack.Install (nodes);

Ipv4AddressHelper address;
address.SetBase ("192.168.0.0", "255.255.0.0");
interfaces = address.Assign (devices);
```

Figure 13: Internet Stack User Code

routing algorithm is used with default parameters in our simulations. The implementation details of the OLSR routing algorithm can be found in NS-3 documentation [67].

AODV routing algorithm is a traditional reactive routing algorithm whose fundamental mechanism is explained in Section 2.2.2. As for the AODV routing protocol, the implementation of the NS-3 simulator is used with default parameters. The implementation details of the AODV routing algorithm can be found in NS-3 documentation [68].

AntHocNet routing algorithm is a bio-inspired routing algorithm whose fundamental mechanism is explained in Section 2.2.3.2. In this research, the AntHocNet routing algorithm is implemented for the NS-3 simulator using the implementation and explanation of Ducatelle as a base [41]. The details of Ducatelle's implementation of AntHocNet routing algorithm for Qualnet4.0 network simulator can be found in [41] and [69].

4.2.2.5 Network Application

As for the application layer, in order to generate constant bit rate traffic, NS-3 *OnOffApplication* module is used. In simulations, one of the nodes is selected as the base node. The base node is stationary and placed at the bottom of the center point of simulation space, at the XYZ coordinates of (1000, 1000, 100). The base node is static and is configured as the sink and the

destination of all CBR traffic. After the simulation starts, ten of the nodes other than the base node are selected randomly to generate CBR traffic throughout the simulation time.

```
1  double SimTime = p_config->totalTime; //120.00;
2  double SinkStartTime = 1;                    //1.0001;
3  double SinkStopTime  = SimTime - 1;          //119.90001;
4  uint16_t port = 9; //RFC 863, Discard Port
5  double app_duration_s = SimTime;
6  int num_of_cbr_traffic_changes = floor(SimTime/app_duration_s); // 1
7  Ptr<UniformRandomVariable>
   ↪  rand_start_time=CreateObject<UniformRandomVariable>();
8  rand_start_time->SetAttribute ("Min", DoubleValue (0));
9  rand_start_time->SetAttribute ("Max", DoubleValue (5));
10
11 for(int i = 0; i < num_of_cbr_traffic_changes; i++){
12   double AppStartTime = SinkStartTime + (i *
     ↪  app_duration_s);
13   double AppStopTime = AppStartTime + app_duration_s;
14   for(uint16_t src_node = 0; src_node < 10; src_node++){
15     Ptr<Node> dst_node = nodes.Get(BASE_NODE_IND);
16           Ptr<Ipv4> ipv4 = dst_node->GetObject<Ipv4> ();
17           Ipv4InterfaceAddress ipv4_int_addr = ipv4->GetAddress (1, 0);
18           Ipv4Address ip_addr = ipv4_int_addr.GetLocal ();
19           OnOffHelper onoff ("ns3::UdpSocketFactory", InetSocketAddress
             ↪  (ip_addr, port));
20           onoff.SetConstantRate (DataRate (p_config->dataRate),
             ↪  p_config->pktSize);
21           ApplicationContainer apps = onoff.Install (nodes.Get
             ↪  (src_node));
22
23           double random_time = rand_start_time->GetValue ();
24           apps.Start (Seconds (AppStartTime + random_time));
25           apps.Stop (Seconds (AppStopTime));
26   }
27 }
```

Figure 14: Network Application Layer NS-3 Code Block

The application layer logic is implemented as shown in Figure 14 in order to generate network load properly. Note that the application in each node randomly starts sending packets within the first 5 seconds of the application duration period.

In the NS-3 simulator, all of the mentioned simulation components should be created and installed before the simulator is started, as shown in Figure 15.

```
1  SeedManager::SetSeed (p_config->seed);
2  SeedManager::SetRun(p_config->runNumber);
3
4  CreateNodes (); //Creates Nodes
5  CreateDevices (); //Create Network Stack
6  AllocateNodes(); //Create and Set Mobility Model
7  InstallInternetStack (); //Create and install Routing Algortihm
8  InstallApplications (); //Create and install Network Application
9
10 std::cout << "Starting simulation for " << p_config->totalTime << " s
   ↪  ...\n";
11 std::cout << "Starting simulation with DataRate : " << p_config->dataRate
   ↪  << " bps ...\n";
12 std::cout << "Starting simulation with " << p_config->numOfNodes << "
   ↪  nodes ...\n";
13 std::cout << "Starting simulation with NodeSpeed : " <<
   ↪  p_config->nodeSpeed << " m/s ...\n";
14 std::cout << "Starting simulation with Routing Type : " <<
   ↪  p_config->routingType << " ...\n";
15 std::cout << "Starting simulation with Mobility Type : " <<
   ↪  p_config->mobilityType << " ...\n";
```

Figure 15: Simulation Components Creation

4.2.3 Data Collection

Data collection from NS-3 simulations is mainly built upon *FlowMonitor* [70]. FlowMonitor is a network monitoring framework for NS-3 that detects the communication flows within the network and stores data concerning well-defined metrics.

The metrics that the FlowMonitor provides are as follows [70]:

- *timeFirstTxPacket* : The time when the first packet of the flow transmitted.
- *timeLastTxPacket* : The time when the last packet of the flow transmitted.
- *timeFirstRxPacket* : The time when the first packet of the flow received.
- *timeLastRxPacket* : The time when the last packet of the flow received.
- *delaySum* : Sum of all end-to-end delays for all received packets.
- *jitterSum* : Sum of all end-to-end jitters for all received packets.
- *txBytes* : Total number of transmitted bytes.
- *txPackets* : Total number of transmitted packets.
- *rxBytes* : Total number of received bytes.
- *rxPackets* : Total number of received packets.
- *lostPackets* : Total number of packets that are assumed to be lost.
- *timesForwarded* : For all received packets, sum of the number of hops made.
- *delayHistogram* : Histogram for packet delays
- *jitterHistogram* : Histogram for packet jitters
- *packetSizeHistogram* : Histogram for packet sizes.

By using the metrics provided by *FlowMonitor*, the data necessary to answer our research questions are obtained. In this experimental research, all the dependent variables are calculated for each run separately. Then, the mean values of the selected dependent variables are calculated for all the runs. As mentioned in Section 4.1, dependent variables are end-to-end delay, end-to-end jitter, packet loss ratio, routing overhead, and mean hop count.

In this research;

- *End-to-End Delay* is calculated using *delaySum*, which contains the sum of all end-to-end delay values for all received packets for each flow, and *rxPackets*, which contains the total number of received packets for each flow.

- *End-to-End Delay* variation is calculated using *jitterSum*, which contains the sum of all end-to-end jitter values for all received packets for each flow, and *rxPackets*, which contains the total number of received packets for each flow. The jitter is defined by *FlowMonitor* in accordance with *Type-P-One-way-ipdv-jitter* defined in RFC3393 [71].

- *Packet Loss Ratio* is calculated using *lostPackets*, which contains the total number of packets that are transmitted but not received or forwarded, and *rxPackets*, which contains the total number of received packets for each flow.

- *Routing Overhead* is obtained by counting the routing packets received successfully by nodes.

- *Mean Hop Count* is calculated using *timesForwarded*, which contains how many times a packet is forwarded for all received packets in each flow, and *rxPackets*, which contains the total number of received packets for each flow.

Although flow monitor provides a solid framework for data collection, some issues should be pointed out. First, the flow monitor marks packets as lost packets if a transmitted packet is not received for 10 seconds by default. Therefore, the packets transmitted within the last 10 seconds of simulation may be marked as lost packets and hence disturb the statistics. In order to overcome this issue, NS-3 simulation time is increased by 10 seconds even if the applications are not running for the last 10 seconds, as shown in Figure 16, lines 16-20.

Another point that should be mentioned is that no data is collected in the first 30 seconds of the simulations in this research. This is because, in the first 30 seconds, it is assumed that routing protocols might not converge and are in the process of learning the network topology. In this research, it is not intended to observe the transient behavior of routing protocols. Therefore, in the first 30 seconds of the simulations, network traffic data is not collected and processed. This can be seen in Figure 16, lines 1 and 9.

```
Time data_col_start_time = Seconds(30.0);
Ptr<FlowMonitor> flowmon; FlowMonitorHelper flowmonHelper;

flowmon = flowmonHelper.InstallAll();
flowmon->SetAttribute("DelayBinWidth", DoubleValue(0.001));
flowmon->SetAttribute("JitterBinWidth", DoubleValue(0.001));
flowmon->SetAttribute("PacketSizeBinWidth", DoubleValue(1));
flowmon->SetAttribute("StartTime", TimeValue(data_col_start_time));
flowmon->CheckForLostPackets();

/*Tracing*/
olsrDataCollectionStartEvent = Simulator::Schedule(data_col_start_time,
↪  &RoutingSimulatorBase::StartOlsrDataCollection, this);
Config::Connect("/NodeList/*/$ns3::olsr::RoutingProtocol/Rx",
↪  MakeCallback (&OlsrRxCallBack));
Config::Connect("/NodeList/*/$ns3::olsr::RoutingProtocol/Tx",
↪  MakeCallback (&OlsrTxCallBack));

//Stop the Simulation after before actual application End time
Simulator::Stop (Seconds (p_config->totalTime + 15));
Simulator::Run ();

Ptr<FlowMonitorStatsReader> flowReader= new FlowMonitorStatsReader();
flowReader->ReadFlowMonitorStats(flowmon, p_config->pktSize,
↪  &olsr_msg_info, p_config->runNumber);

stringstream ss; ss << "flow_reader_stats_" << p_config->runNumber;
flowReader->SaveStats(ss.str());
stringstream ss1; ss1 << "output-data/flow_" << p_config->runNumber <<
↪  ".xml";
flowmon->SerializeToXmlFile(ss1.str(), true, true);

Simulator::Destroy ();
```

Figure 16: Data Collection and Simulator Execution

Lastly, it should also be mentioned that the flow monitor is unable to trace NS-3 OLSR implementation. In order to overcome this issue, the NS-3 tracing mechanism is used together with the flow monitor as shown in Figure 16, lines 10-14. By using NS-3 tracing callbacks, the number of RX packets and RX bytes and the number of TX packets and TX bytes for the OLSR routing algorithm are obtained as shown in Figure 17.

```
1  typedef struct{
2    uint32_t rx_packet_count;
3    uint32_t tx_packet_count;
4    uint32_t rx_byte_count;
5    uint32_t tx_byte_count;
6  }S_OLSR_MSGS_INFO;
7  static S_OLSR_MSGS_INFO olsr_msg_info;
8
9  static void OlsrRxCallBack(std::string context, const PacketHeader &
   ↪  header, const MessageList & messages){
10   if(!is_olsr_datacol_enabled){
11           return;
12   }
13   olsr_msg_info.rx_packet_count += messages.size();
14   olsr_msg_info.rx_byte_count += header.GetPacketLength();
15 }
16 static void OlsrTxCallBack(std::string context, const PacketHeader &
   ↪  header, const MessageList & messages){
17   if(!is_olsr_datacol_enabled){
18           return;
19   }
20   olsr_msg_info.tx_packet_count += messages.size();
21   olsr_msg_info.tx_byte_count += header.GetPacketLength();
22 }
```

Figure 17: OLSR Trace Callbacks

In order to collect the raw data and write to *.txt files*, *FlowMonitorStatsReader* is implemented and used as shown in Figure 16, lines 22-25. In this module, several metrics are calculated for each flow generated by each *OnOffApplication* according to the formulas given in [70]. The only difference is that flows are weighted according to the number of transmitted packets in

the flows, and the weighted average of the calculated metrics is saved to files by *FlowMonitorStatsReader* as shown in Appendix C.

4.2.4 Simulation Scenarios

In order to obtain relevant data to find an answer to the research questions, appropriate simulation scenarios are designed. In this section, these scenarios are stated in detail.

The main distinctive characteristics of FW-UAV Networks are the higher node speed and the movement pattern of nodes. In [12], the typical node speed of an FW-UAV is given as 100 km/h (27.7 m/s). In [72], average flight speed of *Skywalker FW-UAV* is given as 15 m/s. In [15], it is stated that UAVs can have a speed of 30 to 460 km/h (8.3 to 127.7 m/s). By taking these into account, the node speed is selected to range from 5 m/s to 120 m/s. Also, mobility models are selected to fit FW-UAVs such as EGM and PPRZM. Moreover, in all of the scenarios presented in this section, mean end-to-end delay, mean end-to-end delay variation, mean hop count, routing overhead, and packet loss ratio metrics are used as dependent variables to answer the research questions. Also, each scenario is executed with different routing algorithms, namely AODV, OLSR, and AntHocNet, in order to be able to compare these routing algorithms with the selected metrics.

4.2.4.1 Scenario-I Effect of Topology Change Rate

This scenario focuses on the performance of compared routing algorithms in highly dynamic network topologies, one of FW-UAV Networks' main characteristics. Hence, in this scenario, the routing algorithm, the mobility model, and the node speed are used as independent variables to observe the dependent variables in a scenario fitting to FW-UAV Network. More specifically, for each combination of routing algorithm and mobility model, node speeds ranging from 5 to 120 m/s are covered. The Independent and dependent parameters used in this scenario are listed in Table 3

Specifically, with Scenario-I, It is possible to observe the effect of node speed and mobility model on the performances of the compared routing algorithms. Since the node speed and the

Table 3: Parameters Used in Scenario-I

Scenario-I Parameters	
Independent Parameters	
Node Speed	{5, 10, 20, 30, 50, 70, 90, 120}
Mobility Models	{EGM, PPRZM, RWP}
Routing Algorithms	{AODV, OLSR, AntHocNet}
Constants	
Node Number	50
Simulation Space	{2000, 2000, 1500} m
Transmission Range	600 m
CBR Data Rate	8192 bps
Dependent Parameters	
End-to-end Delay	$delaySum/total\_rx\_packets$
End-to-end Jitter	$jitterSum/total\_rx\_packets$
Mean Hop Count	$1 + total\_forwarded\_packets/total\_rx\_data\_packets$
Packet Loss Ratio	$total\_lost\_packets/(total\_rx\_packets + total\_lost\_packets)$
Routing Overhead	$total\_rx\_packets - total\_rx\_data\_packets$

mobility model determine the topology change rate, data to answer the first research question can be collected from Scenario-I.

4.2.4.2 Scenario-II Effect of the Network Load

In this scenario, the data rate of CBR traffic is used as an independent parameter together with node speed and routing algorithm to collect data for observing the effect of network load on the dependent variables. The independent and dependent parameters are listed in Table 4

Table 4: Parameters Used in Scenario-II

Scenario-II Parameters	
Independent Parameters	
Node Speed	{5, 10, 20, 30, 50, 70, 90, 120}
CBR Data Rate	{4096, 16384, 65536} bps
Routing Algorithms	{AODV, OLSR, AntHocNet}
Constants	
Node Number	50
Simulation Space	{2000, 2000, 1500} m
Transmission Range	600 m
Mobility Model	{EGM, PPRZM, RWP}
Dependent Parameters	
End-to-end Delay	$delaySum/total\_rx\_packets$
End-to-end Jitter	$jitterSum/total\_rx\_packets$
Mean Hop Count	$1 + total\_forwarded\_packets/total\_rx\_data\_packets$
Packet Loss Ratio	$total\_lost\_packets/(total\_rx\_packets + total\_lost\_packets)$
Routing Overhead	$total\_rx\_packets - total\_rx\_data\_packets$

Specifically, with Scenario-II, it is possible to observe the effect of the changing network load on the performances of the compared routing algorithms. Network load is adjusted by

changing the packet emission intervals. By executing this scenario, necessary data to answer the second research question can be obtained.

CHAPTER 5

EXPERIMENTAL RESULTS

In this chapter, simulation results are presented and discussed for scenarios explained in Section 4.2.4.

5.1 Results for Scenario-I Effect of Topology Change Rate

This section presents the experimental results for Scenario-I, where the node speed, routing algorithm, and mobility model change as independent parameters, and the results are analyzed. All of the simulations conducted for Scenario-I use the same application and MAC layers. Hence, the simulation results provide the routing algorithm performances in different mobility models and node speeds.

5.1.1 Experimental Results and Analysis

Each result obtained by conducting the simulations for Scenario-I is presented and analyzed in the respective sections.

5.1.1.1 Packet Loss Rate vs. Node Speed

Packet Loss Ratio, or Packet Loss Rate, is an essential indication of routing algorithm performance and a Quality of Service metric. For Scenario-I, the packet loss rates of compared routing algorithms according to the node speed are presented in Figure 18.

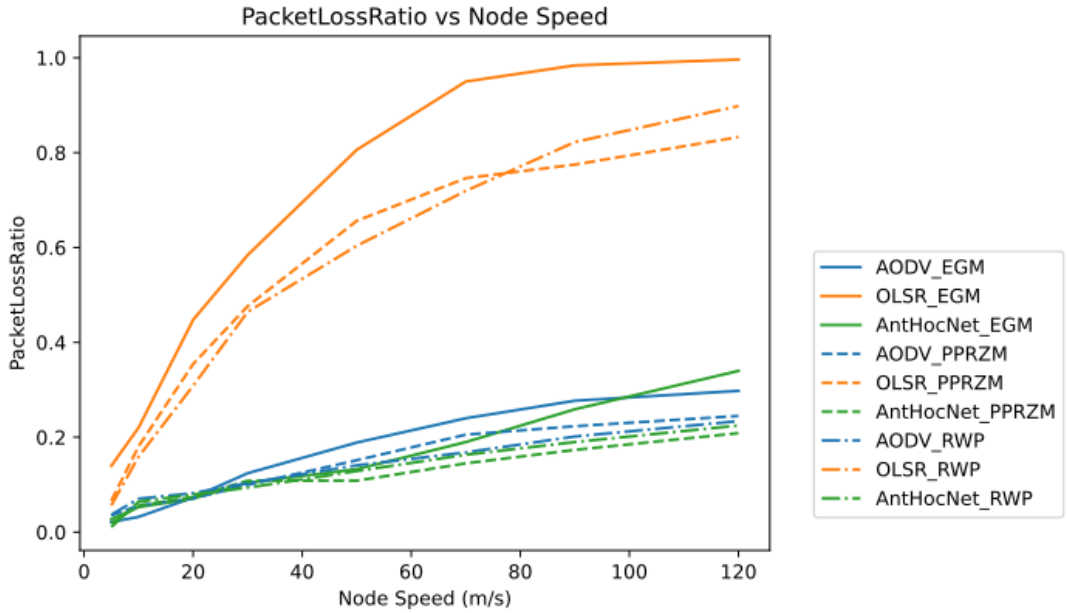


Figure 18: Scenario-I Results: Packet Loss Rate vs. Node Speed

The results in Figure 18 show that the traditional proactive routing algorithm, namely OLSR, cannot cope with the increasing topology change rate. As the topology change rate increases with node speeds, the packet loss ratio for OLSR increases dramatically. This fact stems from the fact that OLSR is a proactive routing protocol. Since the OLSR needs periodic routing messages to notice a change in topology, it gets harder to notice the topology change as the topology change rate increases with increasing node speed. Although it is possible to reduce the period of the OLSR routing messages to notice a change in network topology earlier, this would significantly increase the routing overhead. On the other hand, in AODV and AntHocNet, the increase in packet loss ratio as node speed increases is not as dramatic as that of OLSR. In other words, AODV and AntHocNet are better at detecting a change in network topology than OLSR.

The results of AODV and AntHocNet indicate that the AntHocNet routing algorithm performs slightly better than AODV in the packet loss ratio for PPRZM and RWP mobility models. For the EGM mobility model, only for very high node speeds, i.e., 120 m/s, AODV seems to perform slightly better than AntHocNet in packet loss ratio.

Another point that should be mentioned is that in the EGM mobility model, as the node speed increases, the packet loss ratio for OLSR routing protocol gets closer to 1, indicating that almost all transmitted packets are lost, i.e., no connectivity.

5.1.1.2 Mean Hop Count vs. Node Speed

Mean hop count is an important performance indicator for a routing algorithm. In Scenario-I, mean hop count results are obtained, as shown in Figure 19.

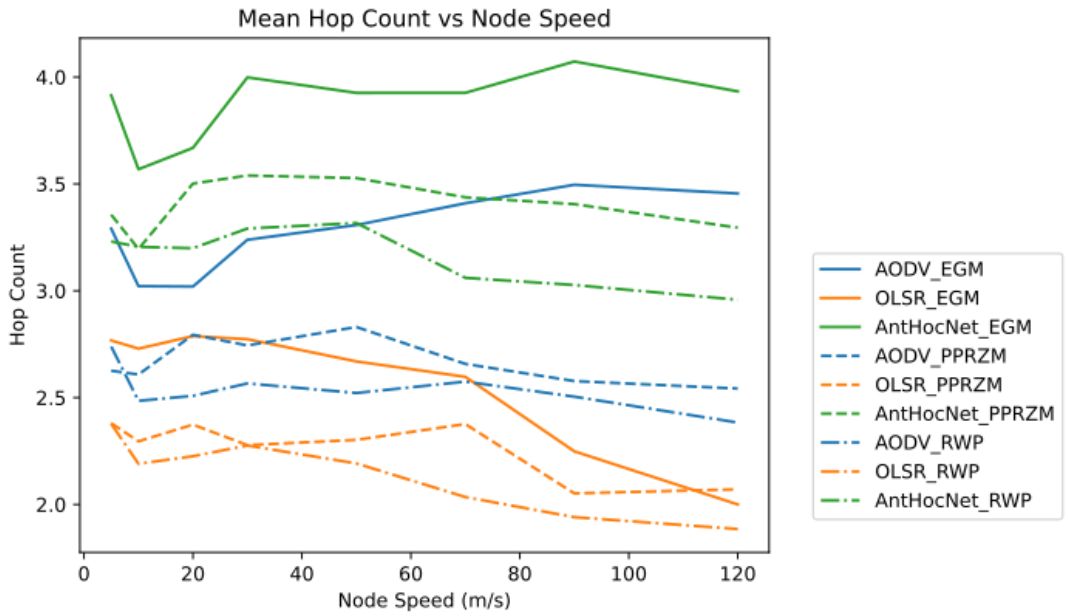


Figure 19: Scenario-I Results: Mean Hop Count vs. Node Speed

If the hop count results are considered together with the packet loss ratio results, it can be seen that the AntHocNet routing algorithm can maintain longer routing paths than AODV and OLSR; because it is observed that even though the packet loss rate for AntHocNet is lower than AODV and OLSR, mean hop count is higher. On the other hand, the OLSR routing algorithm cannot maintain longer paths as node speed increases; because it can be seen that mean hop count decreases and packet loss rate increases as node speed increases. This behavior indicates that, as node speed increases, the OLSR routing algorithm cannot maintain multi-hop routing, and only one-hop or two-hop routing paths are successfully used in packet delivery. Because as the routing path length, i.e., number of hops, increases, the

possibility of being affected by the change of topology increases. Furthermore, OLSR is affected by this situation more than AODV and AntHocNet because of its proactive nature.

5.1.1.3 End-to-End Delay vs. Node Speed

The measured end-to-end delays for simulations conducted for Scenario-I are presented in Figure 20. These results show that, in terms of mean end-to-end delay, the AntHocNet routing algorithm performs better than AODV and OLSR routing algorithms as the speed of nodes increases independent of which mobility model is used.

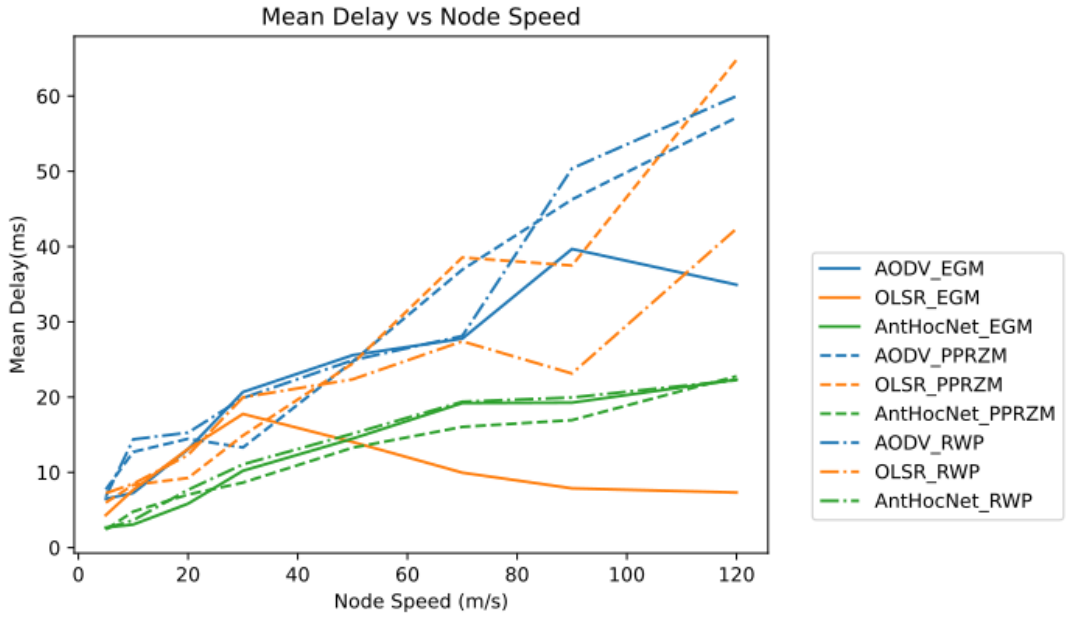


Figure 20: Scenario-I Results: Mean End-to-end Delay vs. Node Speed

First of all, it should be noted that end-to-end delay values increase as the node speed increases for all selected routing algorithms with all the selected mobility models, except the OLSR with the EGM mobility model. The main reason behind this situation stems from the route maintenance process. As the topology change rate increases, the number of link failures increases. Consequently, the need to execute the route maintenance process also increases. Hence, the time spent in the route maintenance process increases, which causes the end-to-end delay measurements to increase.

In simulation results, the mean delay value for the OLSR routing algorithm in the EGM mobility model seems to be lower than AntHocNet; however, this does not indicate that OLSR provides lower end-to-end delay measurements than that of AntHocNet. Packet loss rate approaches to one as node speed increases for OLSR routing protocol with EGM mobility model as shown in Figure 18. Furthermore, as it is shown in Figure 19, the mean hop count of the OLSR routing protocol in the EGM mobility model decreases significantly as the speed of nodes increases. These two results indicate that as node speed increases, OLSR routing protocol cannot maintain multi-hop connectivity between nodes, and multi-hop communication degrades. In other words, most of the packets forwarded through multi-hop paths are lost and are not counted for the mean end-to-end delay, which decreases the mean end-to-end delay value for the OLSR routing protocol. Therefore, it can be said that, apart from this anomaly, AntHocNet provides lower end-to-end delay measurements than AODV and OLSR routing protocols regardless of which mobility model is used.

The main reason AntHocNet can provide lower end-to-end delay measurements is its ability to cope with the change of network topology. Even though AntHocNet executes its route setup process reactively, it also runs a proactive route maintenance process. In other words, different from AODV, AntHocNet aims to find alternate routes before detecting link failures. Thanks to this proactive approach, as the topology change rate increases, the gap between the delay values of AODV and AntHocNet grows; because, in case of broken links, AntHocNet can provide alternate routes more quickly than AODV.

5.1.1.4 End-to-End Delay Variation (Jitter) vs. Node Speed

Figure 21 shows that end-to-end delay variation changes are similar to end-to-end delay changes as the node speed increases.

For the OLSR routing algorithm in the EGM mobility model, an anomaly described in Section 5.1.1.3 is also be observed for jitter. As stated, for this sub-scenario (OLSR routing protocol and EGM mobility model in increasing node speed), most packets do not reach their destinations and are not considered for delay variation. Apart from this anomaly, it can be seen

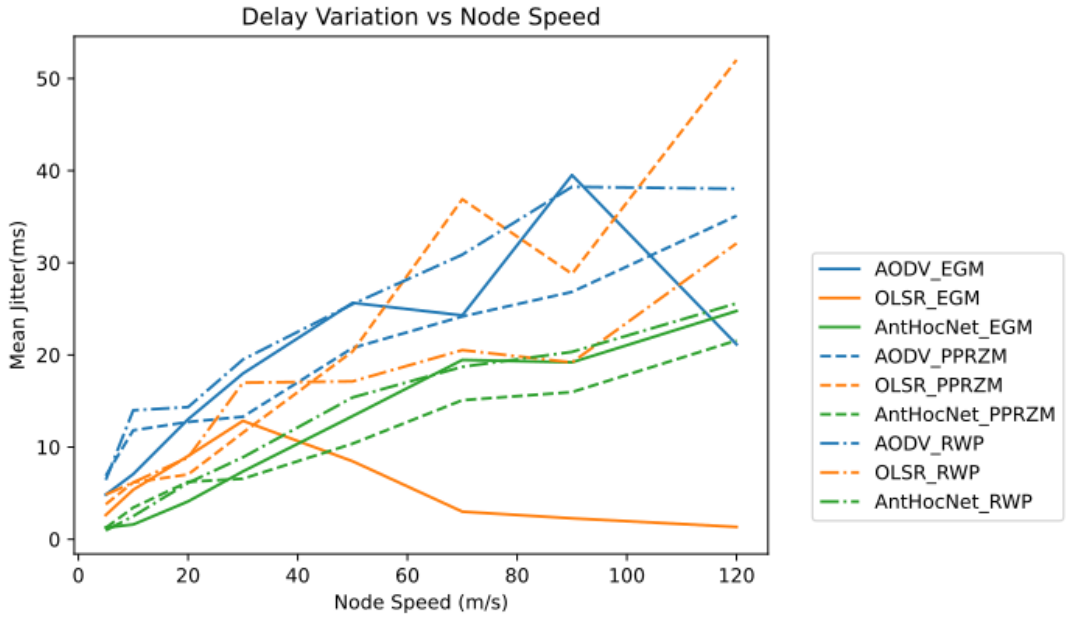


Figure 21: Scenario-I Results: Jitter vs. Node Speed

that the AntHocNet routing algorithm provides better jitter levels than traditional AODV and OLSR routing algorithms.

5.1.1.5 Routing Overhead vs. Node Speed

As stated in Section 4.2.3, routing overhead is measured in terms of the number of routing packets received by nodes. As a proactive routing protocol that periodically sends routing packets to the network, the OLSR routing algorithm naturally has the highest routing overhead regardless of the mobility model, as shown in Figure 22.

According to simulation results conducted for Scenario-I, it can be said that AntHocNet routing protocol provides above mentioned QoS metrics, i.e., end-to-end delay, end-to-end jitter, and packet loss ratio, with a lower number of routing packets than traditional AODV and OLSR routing protocols. Moreover, two points require a further explanation in routing overhead results given in Figure 22;

- Firstly, it is intriguing that as node speed increases, the routing overhead of the OLSR routing protocol decreases. In normal circumstances, independent of the node speed,

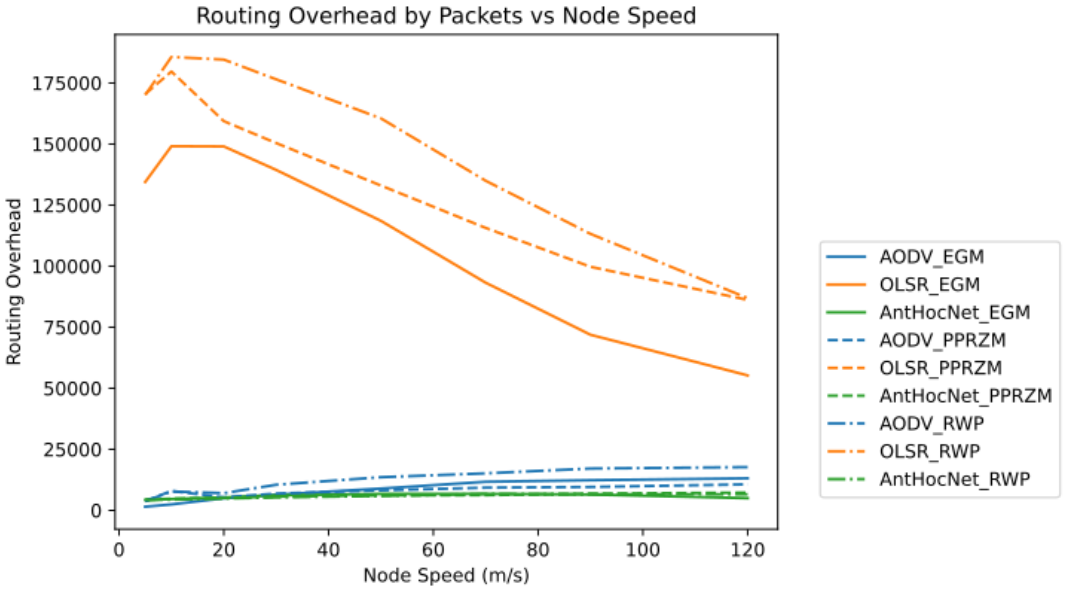


Figure 22: Scenario-I Results: Routing Overhead vs. Node Speed

mobility type, or CBR traffic data rate, OLSR would try to scan the whole network periodically to learn the network topology via hello messages and then disseminate this information to the whole network via TC messages. However, it can be seen from packet loss ratio results that it is hard to achieve this in highly dynamic network topologies for a link-state routing protocol. As topology change rate increases, network packets-either routing or data packets- are lost more and more, and the network becomes partitioned due to missing routing packets. When a routing packet, i.e., TC message, is lost, it cannot trigger subsequent routing packet broadcasting starting from the destination node of the lost routing packet. Because the loss of a routing packet leads to ceased subsequent broadcasts and reduction in the routing overhead. That is, the total received routing packet count decreases when the packet loss rate increases due to the increase in topology change rate. In other words, as node speed increases, more routing packets are lost, and they obstruct routing packet broadcasting at their destination nodes, decreasing the routing overhead but causing routing problems.

- Secondly, as a hybrid routing protocol, it is interesting that AntHocNet, which has proactive components (PFWA and PBA), has a lower routing overhead than AODV at high node speeds. As shown in Table 5, at lower node speeds, AntHocNet routing

overhead is higher than AODV routing overhead, which is expected because of the periodic proactive ant packet transmission. However, as node speed increases, topology change rate increases, resulting in a sharper increase in Routing Overhead for AODV algorithm than that of AntHocNet and causes AODV to have higher routing overhead than AntHocNet in higher node speeds.

AODV protocol is triggered when a data packet is required to be sent. When triggered, AODV searches a route by using RREQ and RREP packets. If no valid route is ready to use for the data packet, then RREQ packets are flooded (or broadcasted) towards the destination node. However, as the topology change rate increases, RERR packets increase due to broken links, triggering new RREQ floods towards the destination node. Because of this mechanism, as the node speed increases, the topology change rate increases, resulting in more and more broken links, and this causes the routing overhead of the AODV protocol to increase significantly.

Similar to the AODV protocol, the AntHocNet protocol is also triggered when a data packet needs to be sent. The AntHocNet route setup process is very similar to the AODV route setup process. The main difference between them is their *route maintenance process*, which starts after the *route setup process* is completed. While AODV has a simplistic route maintenance process that cleans old route information and detects broken links, AntHocNet has a more sophisticated mechanism. In AntHocNet, proactive ants are used to update route information periodically. For the AntHocNet protocol, both in the *route setup process* and the *route maintenance process*, some of the ants are sent in a stochastic way which increases the protocol's flexibility, especially in dynamic network topologies. These properties of AntHocNet provides it to maintain available route information longer with the overhead of proactive ant packets.

The results show that for less dynamic network topologies, AODV protocol provides lower routing overhead since AntHocNet protocol needs to send proactive ants periodically, which increases routing overhead; however, as the topology change rate increases, the routing overhead of the AODV routing algorithm increases more than routing overhead of AntHocNet routing algorithm. This phenomenon shows that the AntHocNet *route maintenance process* can cope with increasing topology change rates better than the AODV *route maintenance process*. Because as the topology change rate increases,

Table 5: Scenario-I Results: AODV and AntHocNet Routing Overhead

AODV and AntHocNet Routing Overhead Results			
Mobility	N.Speed	Routing Overhead (AODV)	Routing Overhead (AntHocNet)
EGM	5	1500	4269
EGM	10	2436	4619
EGM	20	4965	5278
EGM	30	6441	6148
EGM	50	8925	6742
EGM	70	11706	6744
EGM	90	12348	6505
EGM	120	13140	4985
PPRZM	5	3789	4246
PPRZM	10	7945	4792
PPRZM	20	5332	5073
PPRZM	30	6879	5613
PPRZM	50	8185	6120
PPRZM	70	9294	6225
PPRZM	90	9573	6856
PPRZM	120	10647	7239
RWP	5	3731	4280
RWP	10	7752	4389
RWP	20	7033	4851
RWP	30	10483	5296
RWP	50	13576	5890
RWP	70	15157	6718
RWP	90	17121	6601
RWP	120	17699	6597

the AODV protocol has a higher need to re-run the *route setup process*, which includes flooding RREQ packets, while AntHocNet protocol can maintain its routes.

5.2 Results for Scenario-II Effect of the Network Load

In this section, the experimental results for Scenario-II, where the data rate of CBR traffic, node speed, and the routing algorithm are changing as independent parameters, are presented, and results are analyzed. In this scenario, as the data rate increases, the communication load of the network increases. This scenario makes it possible to measure how well the routing algorithms behave under various network loads. In this scenario, the data rate of CBR traffic is selected as either 4096 bps, 16384 bps, or 65536 bps. The simulations are conducted separately for the EGM, PPRZM, and RWP, respectively.

5.2.1 Experimental Results and Analysis

The simulation results for Enhanced Gauss-Markov Mobility Model are given in Figures 23, 26, 29, 33, and 36. The simulation results for Paparazzi Mobility Model are given in Figures 24, 27, 30, 34, and 37. Lastly, the simulation results for RWP Mobility Model are given in Figures 25, 28, 31, 35, and 38. Each of these results is presented and analyzed in the respective sections.

5.2.1.1 Packet Loss Rate vs. Node Speed

The packet loss rates of compared routing algorithms are given according to node speed and data rate in Figures 23, 24, and 25.

First of all, it should be noted that as the CBR data rate increases from 4096 bps to 65536 bps, the packet loss rate increases for the compared routing algorithms, regardless of the node speed and mobility model. This increase in packet loss rate is more prominent for AODV and AntHocNet than for OLSR, as shown in Figures 23, 24, and 25. *The network congestion*,

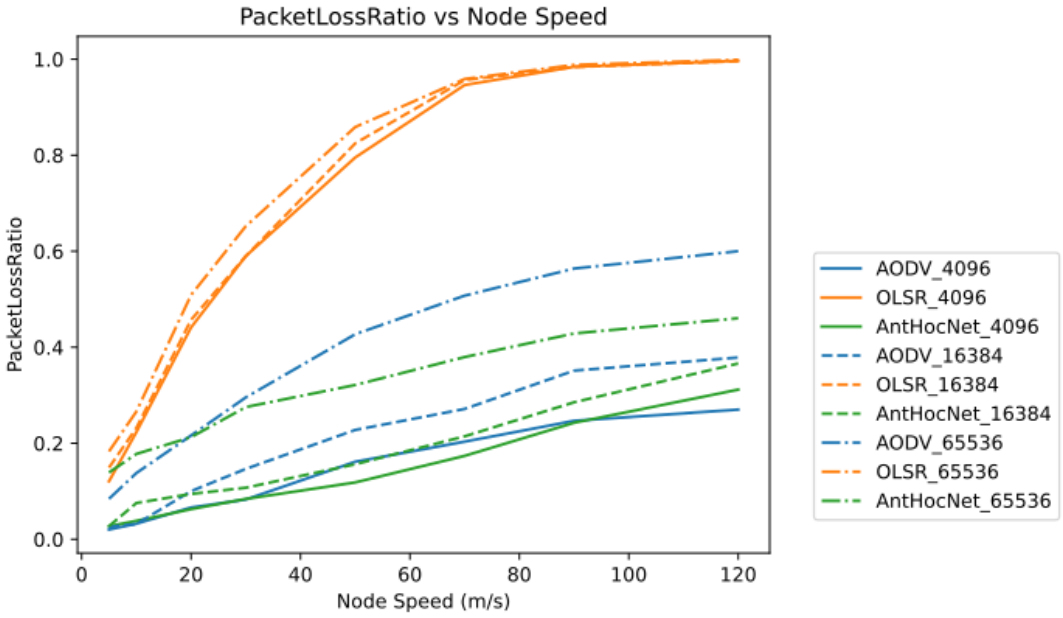


Figure 23: Scenario-II Results: Packet Loss Rate vs. Node Speed (EGM)

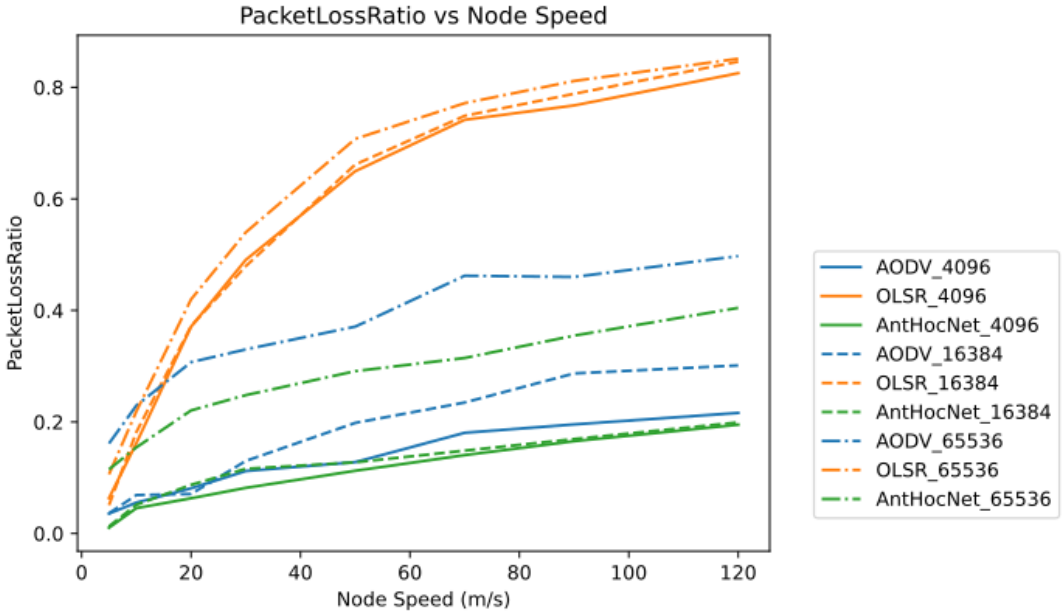


Figure 24: Scenario-II Results: Packet Loss Rate vs. Node Speed (PPRZM)

caused by the data rate increase, seems to be the main reason for the packet loss rate to increase as the data rate increases.

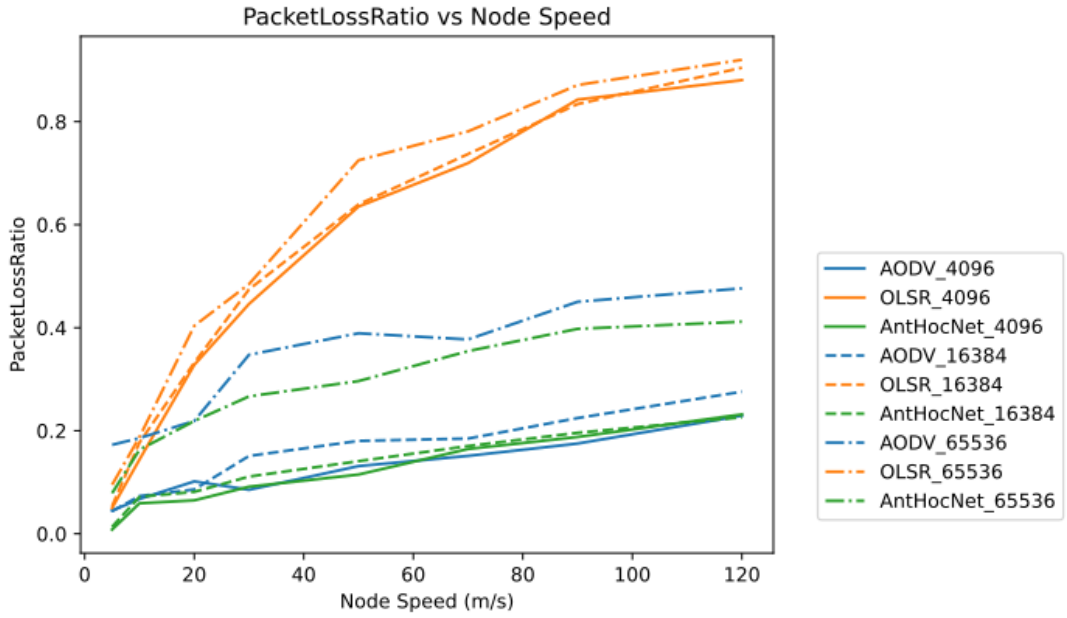


Figure 25: Scenario-II Results: Packet Loss Rate vs. Node Speed (RWP)

Secondly, similar to the results presented in Section 5.1.1.1, as the node speed increases, the packet loss rates of all routing algorithms increase. However, in the OLSR routing algorithm, this increase in packet loss rate is much more dramatic than AODV or AntHocNet routing algorithms. Because as the node speed increases, the possibility of the link failure on the longer paths increases, and it gets much more challenging for OLSR to maintain its routes than AODV and AntHocNet.

In the simulations conducted with the EGM mobility model, the packet loss rate for the OLSR routing algorithm goes towards 1, i.e., no-connectivity region, for all of the selected data rate values. In other words, almost all transmitted packets are lost with the OLSR routing algorithm. However, in simulations conducted with PPRZM and RWP, the packet loss rate for the OLSR routing algorithm increases around 0.8 to 0.85.

Lastly, the AntHocNet routing algorithm can provide a lower packet loss ratio than the AODV and OLSR routing algorithms regardless of the data rate and the mobility model the simulations use. The exception of this observation occurs when the EGM mobility model is used with a 4096 bps data rate. In the exception case, the AODV routing algorithm provides better packet loss ratios at some very high node speeds than the AntHocNet routing algorithm.

5.2.1.2 Mean Hop Count vs. Node Speed

For Scenario-II simulations, the mean hop counts of compared routing algorithms according to node speed and data rate are given in Figures 26, 27, and 28.

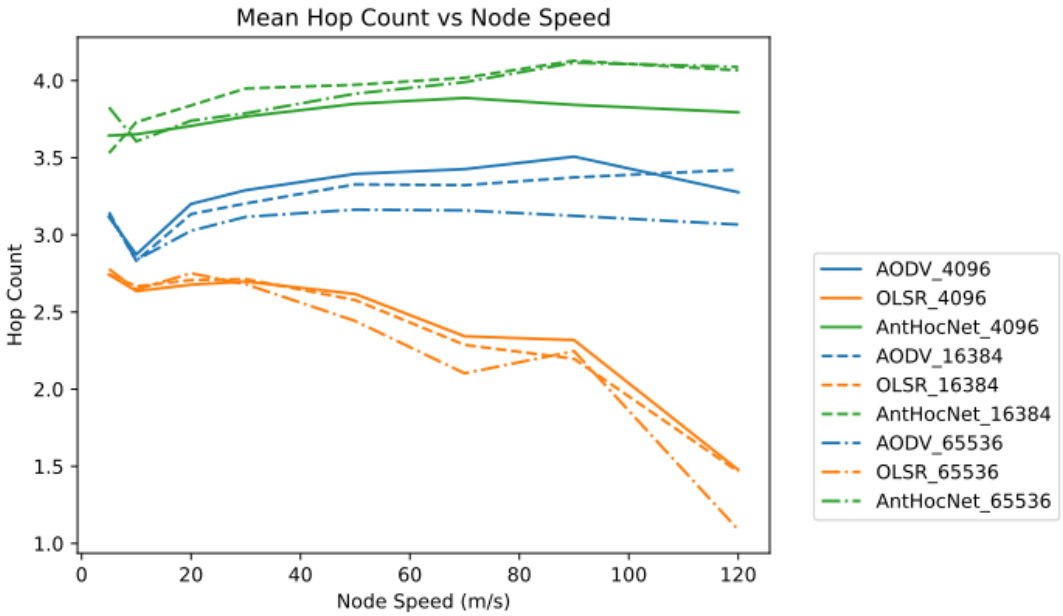


Figure 26: Scenario-II Results: Mean Hop Count vs Node Speed (EGM)

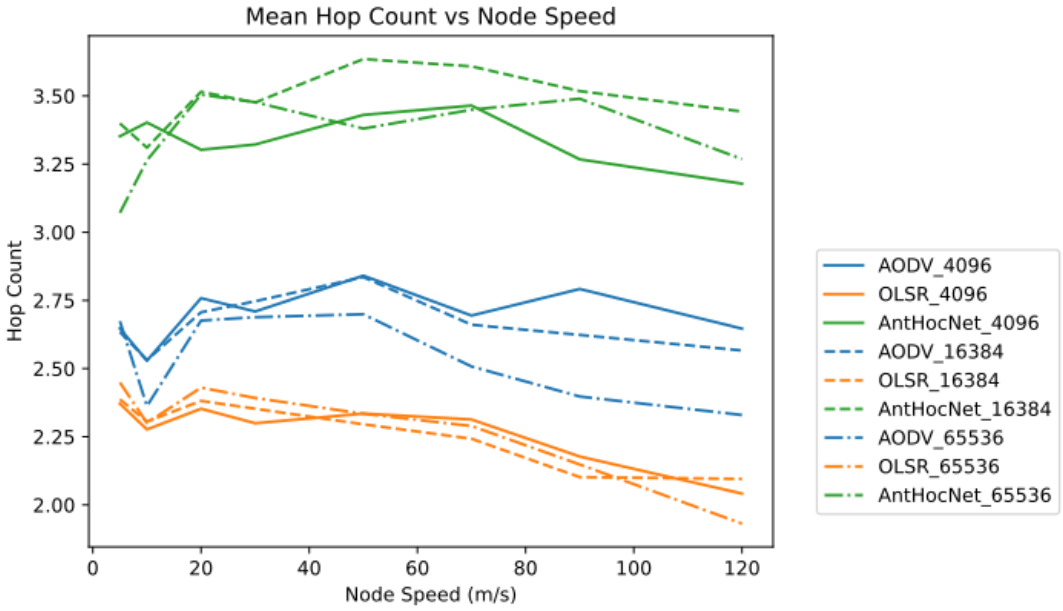


Figure 27: Scenario-II Results: Mean Hop Count vs Node Speed (PPRZM)

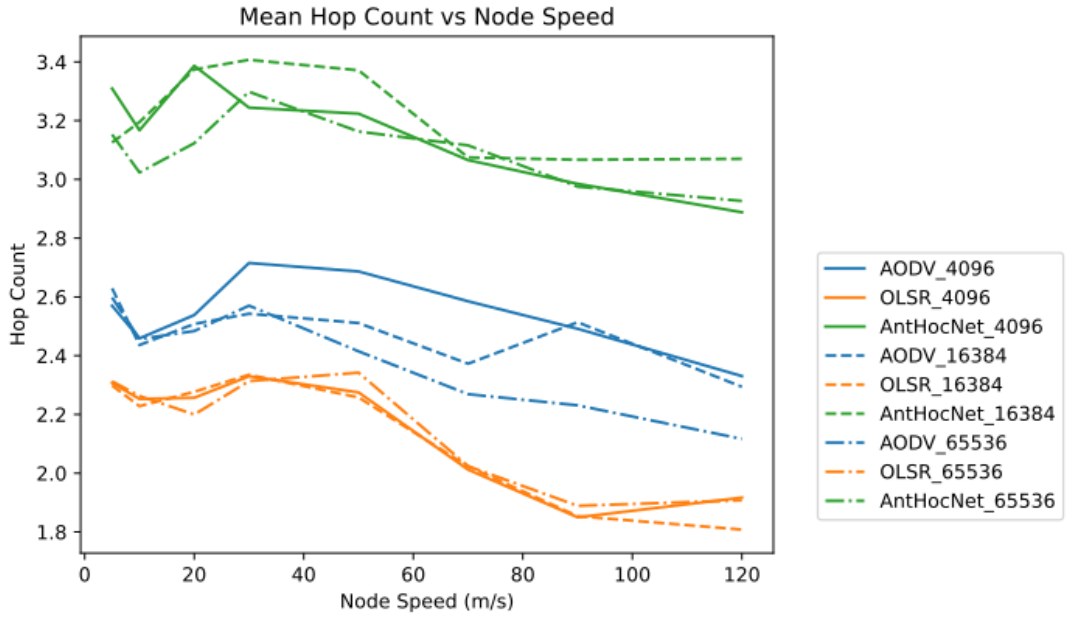


Figure 28: Scenario-II Results: Mean Hop Count vs Node Speed (RWP)

First of all, the results indicate that the data rate does not directly relate to the mean hop counts; however, it is evident that the change in data rate affects the mean hop count. For AODV, it can be said that an increase in the data rate generally decreases the mean hop count. Because the network congestion caused by increasing data rate also affects the routing messages and it gets harder for AODV to maintain longer paths. Even though this can be claimed for AODV, it does not seem to be the case for AntHocNet and OLSR.

Similar to the situation described in Section 5.1.1.2, the mean hop count provided by the OLSR routing algorithm decreases as the node speed increases, regardless of the data rate or the mobility model. However, it should be noted that this decrease in the mean hop count does not indicate that OLSR is better at finding shorter routing paths. On the contrary, this decrease stems from OLSR's inability to find longer paths at high node speeds. Because, as the node speed increases, longer routing paths are becoming more inclined to be broken. In conclusion, the results presented in Figures 26, 27, and 28 indicate that OLSR cannot provide longer routes as the node speed increases regardless of the data rate.

Another point that should be emphasized is that, in all of the simulations, regardless of the data rate and mobility model, the AntHocNet routing algorithm has higher mean hop counts.

When this situation is considered with the mean end-to-end delay results and packet loss rate results, it can be said that AntHocNet provides a more comprehensive multi-hop network; in other words, AntHocNet can find longer paths with higher hop counts than AODV and OLSR. Because, thanks to the ability to find paths with higher hop count, the packet loss rate of the AntHocNet routing algorithm decreases.

5.2.1.3 End-to-end Delay vs. Node Speed

For Scenario-II simulations, mean end-to-end delay values of compared routing algorithms according to node speed and data rate are given in Figure 29, Figure 30, and Figure 31. The results show that the measured end-to-end delay values for AODV, OLSR, and AntHocNet protocols with indicated data rates range from 0 to 80 ms.

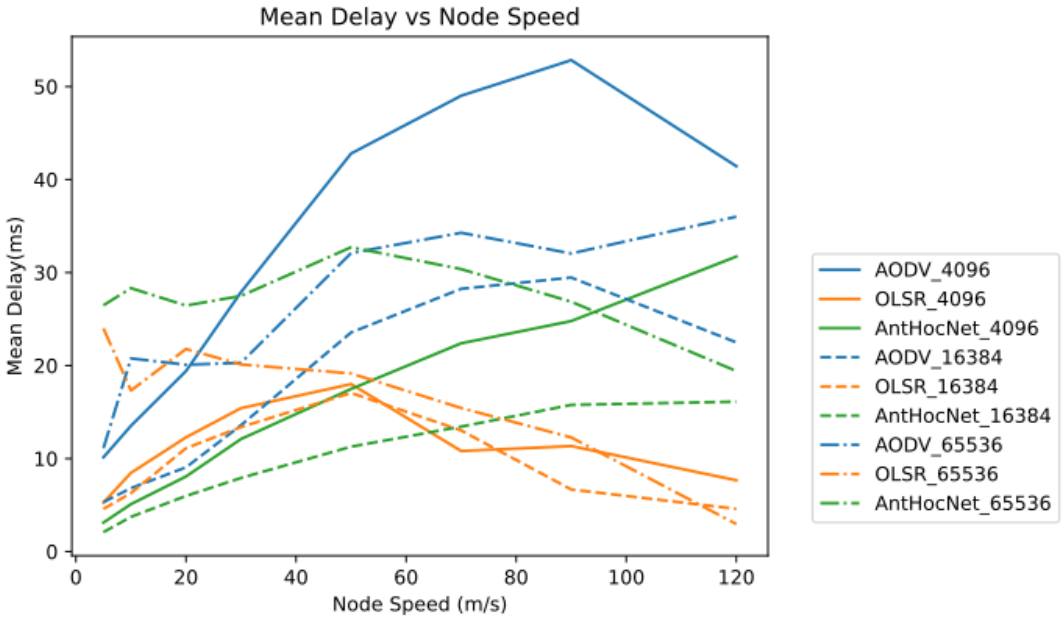


Figure 29: Scenario-II Results: End-to-End Delay vs Node Speed (EGM)

In general, it is expected that as the data rate increases, the measured delay values increase because of the network congestion. Furthermore, it is expected that as the node speed increases, the measured delay values increase due to the increased route maintenance delays. However, when the results are analyzed, it can be seen that these anticipations are not the case. Firstly, it is impossible to observe that the delay measurements increase as the data rate increases.

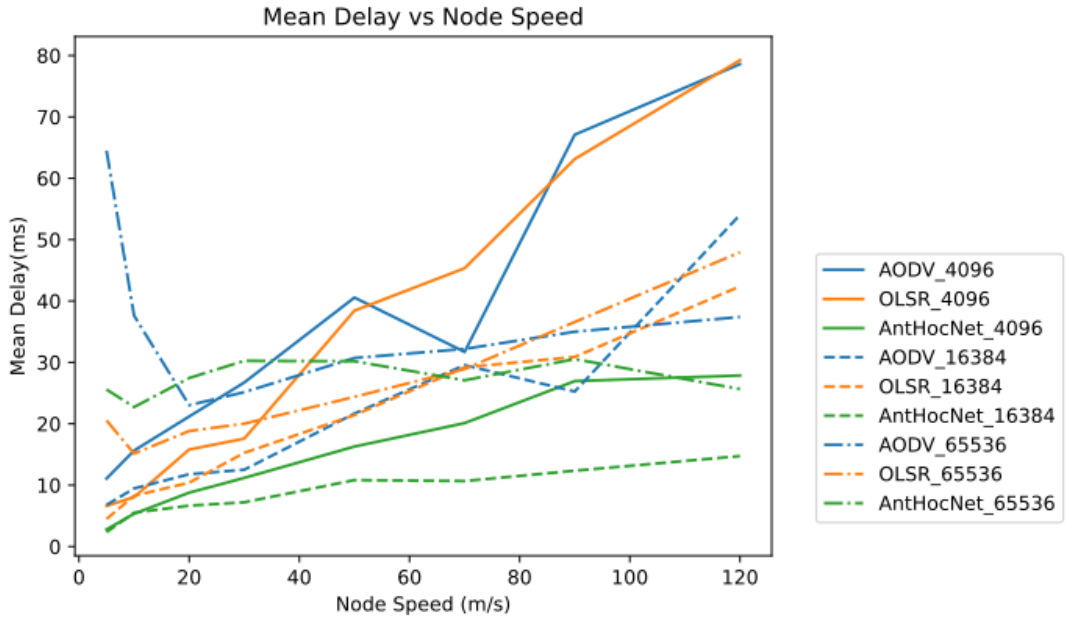


Figure 30: Scenario-II Results: End-to-End Delay vs Node Speed (PPRZM)

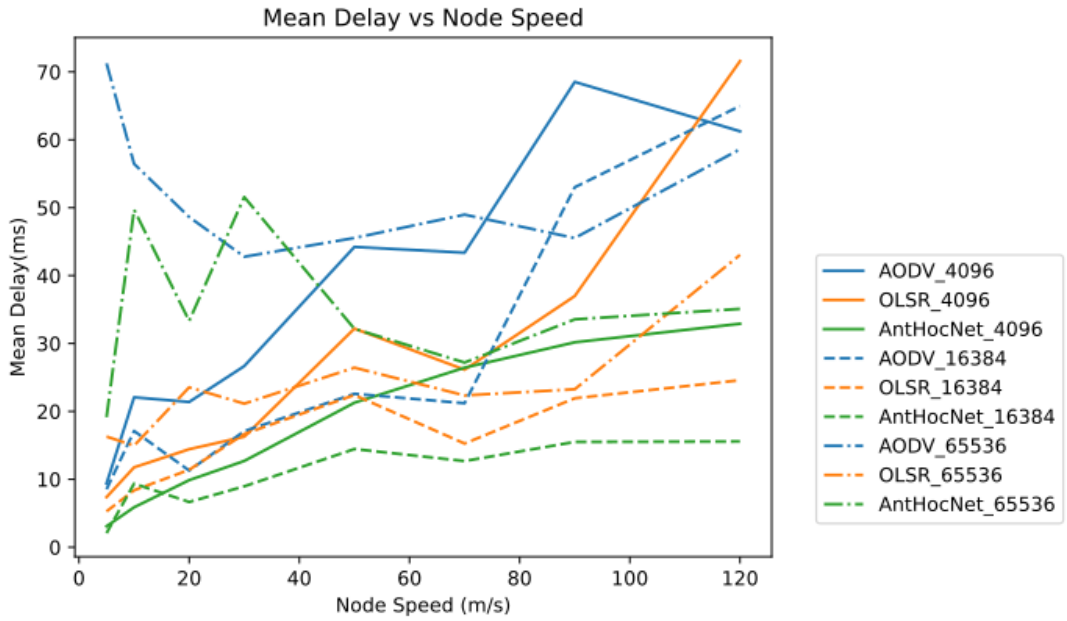


Figure 31: Scenario-II Results: End-to-End Delay vs Node Speed (RWP)

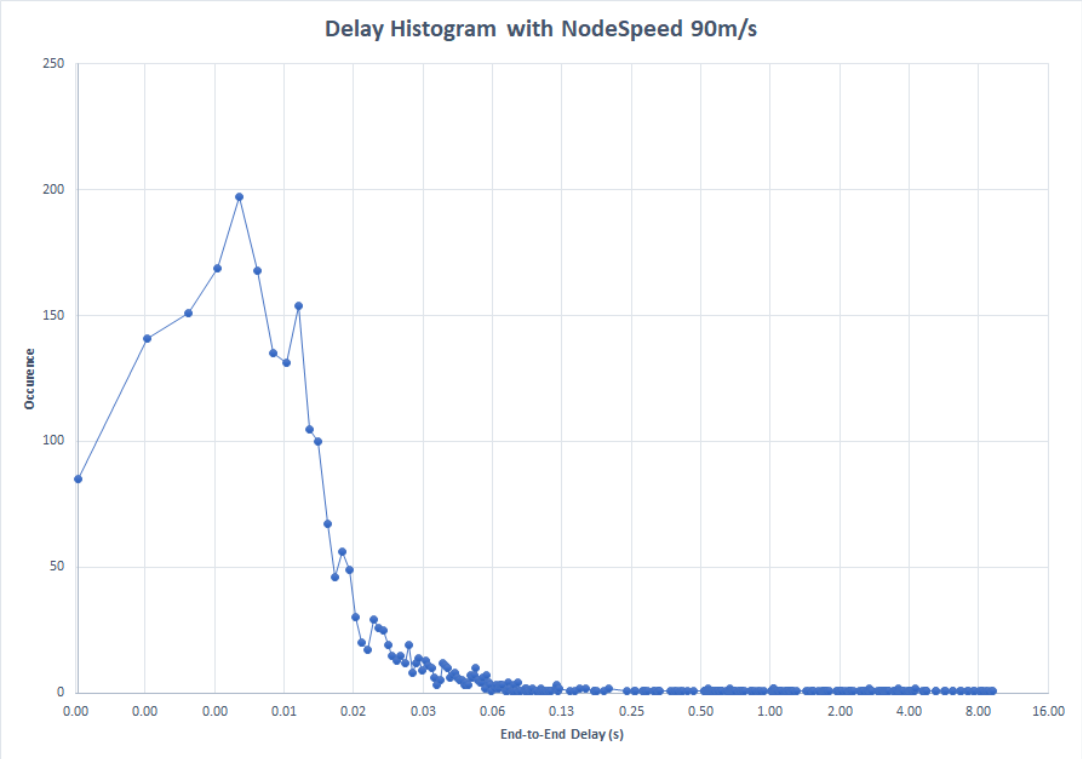
At different topology change rates, different cases with different data rates are providing the highest delay values. Secondly, not all of the delay measurements increase as the node speed increases.

Concerning the anticipation regarding the relationship between data rate and end-to-end delay, it can be said that the effect of data rate is more complicated than anticipated. An increasing data rate has contrasting effects on end-to-end delay measurements. On the one side, the increase in data rate also increases the network congestion and consequently end-to-end delays. On the other hand, an increased data rate also increases packet loss in longer paths and decreases end-to-end delays. Because of this situation, it is hard to see the direct relationship between data rate and mean delay values.

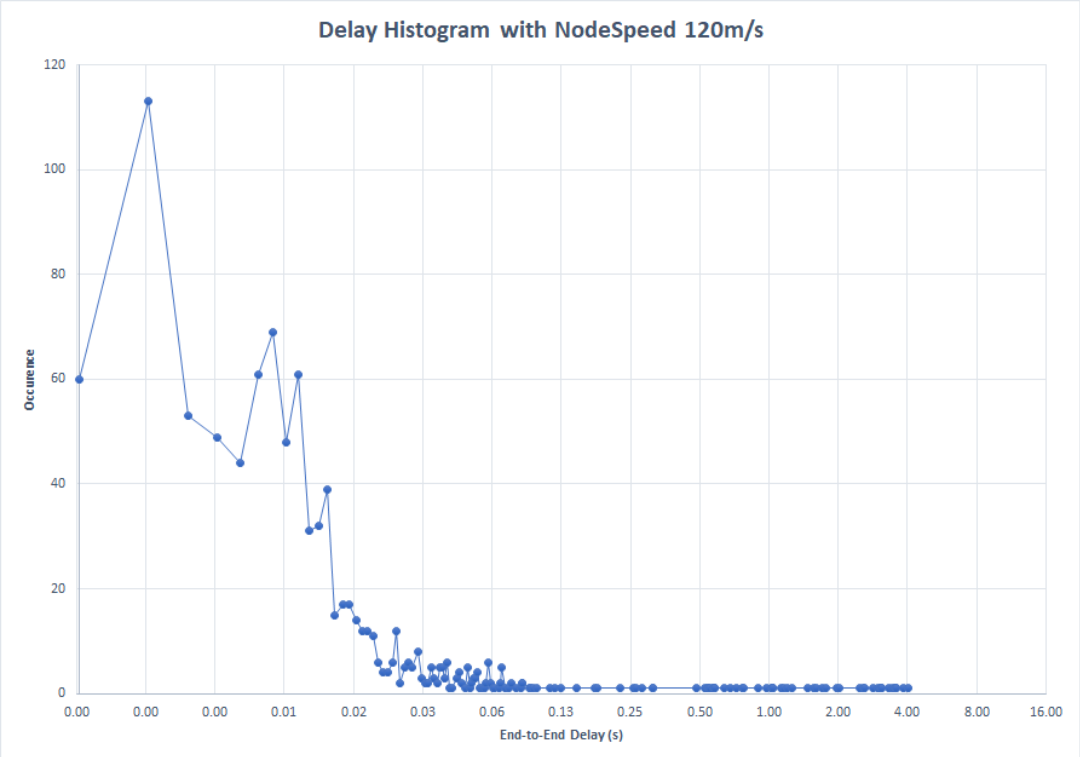
About the anticipation regarding the relation of node speed with end-to-end delay, some points can be explained. Firstly, as node speed increases, the delay value provided by the OLSR routing protocol for all selected data rates decreases when the EGM mobility model is used. Similar to the explanation given in Section 5.1.1.3, because packet loss ratio measured for OLSR routing algorithm closing towards 1 for EGM mobility model, end-to-end delay of the OLSR protocol decreases significantly; because none of the lost packets are taken into account in calculating end-to-end delay values. Because of this situation, OLSR mean delay value decreases as node speed increases when the EGM mobility model is used. On the other hand, when the PPRZM or RWP mobility models are used, the packet loss ratio for the OLSR routing algorithm only rises around 0.85. The data collection mechanism can obtain more samples, which are not limited to the one or two-hop communications. Therefore, the mean end-to-end value calculated for the OLSR routing algorithm for all selected data rates increases as node speed increases.

Secondly, for the AODV routing algorithm with data rate 4096 bps and EGM mobility model, when node speed is 90 m/s, there is a peak in end-to-end delay value. However, when delay histograms for *nodespeed-90m/s* and *nodespeed-120m/s* are inspected, the delay values measured for node speed of 90m/s and node speed of 120m/s do not significantly differ.

The delay histograms for the AODV routing protocol with stated node speeds and a data rate of 4096 bps are presented in Figure 32. For node speed of 90m/s, %87.6 percent of the end-to-end delay measurements (2331 out of 2662) are between 0 to 50 ms. For node speed of 120m/s, %89.4 percent of the end-to-end delay measurements (902 out of 1009) are between 0 to 50 ms. Even though the total transmitted packets, i.e., the data rate and the simulation time, are the same for these two node speeds, the total number of delay measurements are



(a) Scenario-II Results: DelayHistogram when Node Speed is 90m/s



(b) Scenario-II Results: DelayHistogram when Node Speed is 120m/s

Figure 32: Scenario-II Delay Histograms with EGM Mobility Model

different. This situation is because when node speed rises from 90 m/s to 120 m/s, total received packets decrease due to the increased topology change rate, reducing the total number of delay measurements.

As emphasized, even though most measurements range from 0 ms to 50 ms, the mean delay value for node speed of 90 m/s is higher because there are delay measurements around 9 seconds, as shown in Figure 32a. The data for these histograms are given in Appendix D.2.1.1.

When the results are analyzed, it is possible to see that the AntHocNet routing algorithm provides slightly better end-to-end delay values than AODV and OLSR for the PPRZM and RWP mobility models. On the other hand, for the EGM mobility model, it is possible to state that AntHocNet provides better delay values than AODV.

5.2.1.4 End-to-end Delay Variation (Jitter) vs. Node Speed

Results presented in Figures 33, 34, and 35 show that delay variation measurements for the indicated scenarios range from 0 to 70 ms. When the data rate is 4096 bps and the AODV routing algorithm is used, the measured jitter value shows similar behavior to the measured delay value as the node speed increases, as given in Figure 33. These results are in accordance with results obtained for end-to-end delay values presented in Section 5.2.1.3.

Another point is that when the mean end-to-end delay value of the OLSR routing protocol drops significantly due to the packet loss ratio going over 0.9, delay variation values also approach zero for the EGM mobility model. The mechanism behind this situation starts with an increasing topology change rate. OLSR cannot cope with the higher topology change rates, and the packet loss rate increases dramatically. Also, the mean hop count decreases because OLSR cannot maintain longer paths, as explained in Section 5.2.1.2. Like end-to-end delay values, delay variation values are also calculated by using the received packets. Since the received packets are from shorter routes, their delay variation values are small. In other words, the possibility of shorter routes being affected by the topology change is less than that of long paths, which reduces the delay variation. Hence, delay variation values decrease. The delay variation result of this mechanism can be observed from Figure 33 where the jitter for the OLSR decreases as the node speed increases.

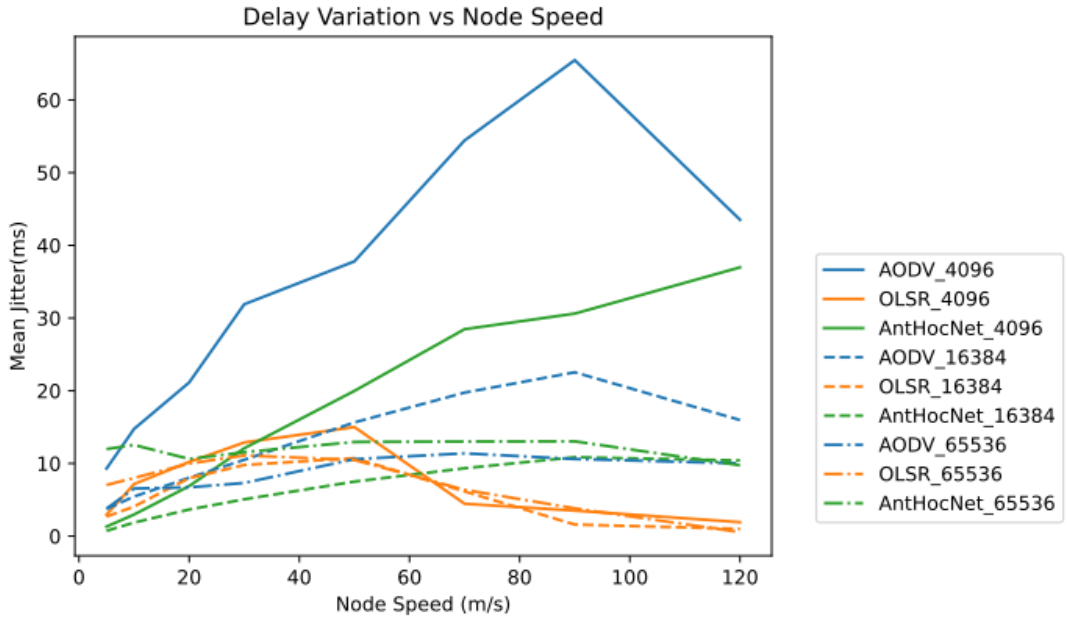


Figure 33: Scenario-II Results: End-to-End Delay Variation vs Node Speed (EGM)

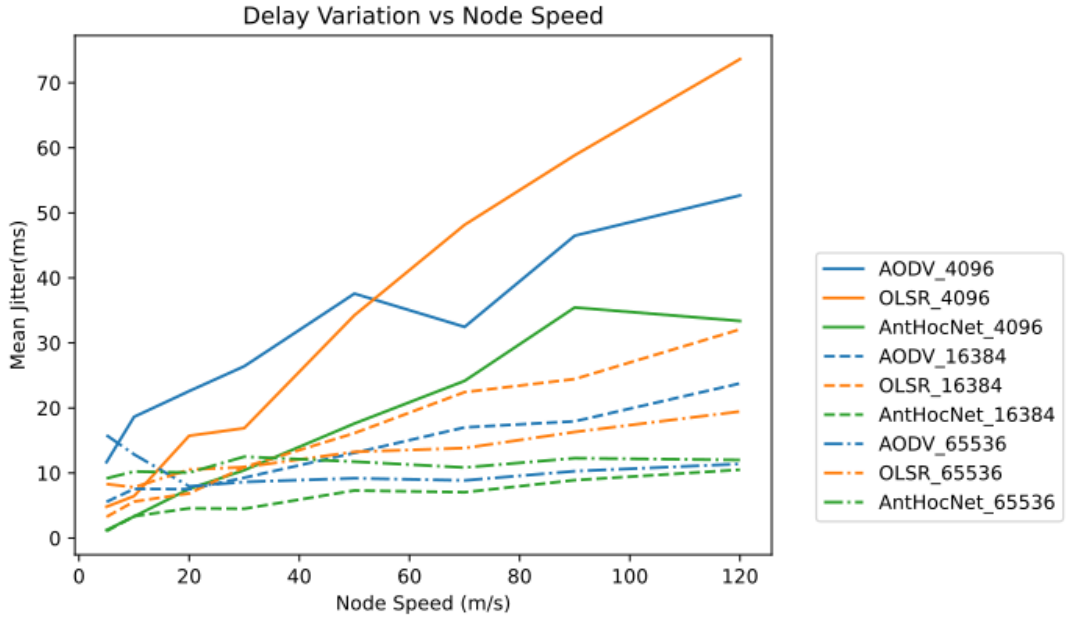


Figure 34: Scenario-II Results: End-to-End Delay Variation vs Node Speed (PPRZM)

Unlike end-to-end delays, it is hard to claim that any of the compared routing protocols are better than others in terms of jitter. While in some cases, AntHocNet is performing better than

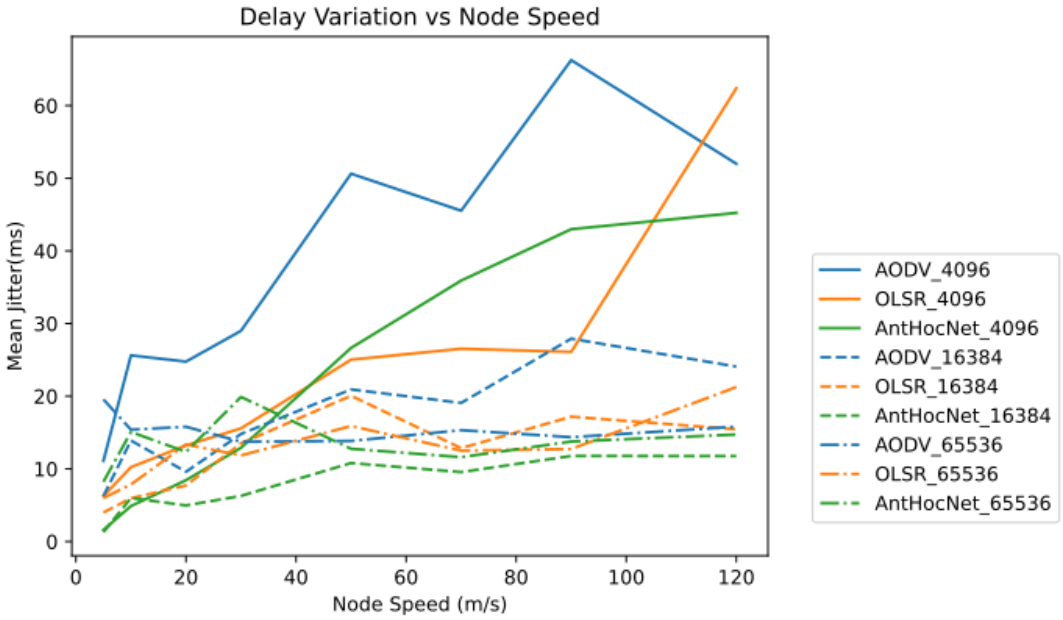


Figure 35: Scenario-II Results: End-to-End Delay Variation vs Node Speed (RWP)

OLSR and AODV in terms of jitter, for instance, when the data rate is 16384bps as shown in Figure 34; in other cases, AODV or OLSR may perform better.

5.2.1.5 Routing Overhead vs. Node Speed

The simulations executed for Scenario-II with EGM, PPRZM, and RWP mobility models are presented in Figures 36, 37, and 38.

Due to that routing overhead indicates the number of routing packets received by the nodes of FANET, it is expected that the OLSR routing protocol causes higher routing overhead than AODV and AntHocNet routing protocols. This expectation is in accordance with the simulation results obtained for all three of the mobility models presented in Figures 36, 37, and 38.

Apart from this, there are some points which require further explanation. These points are given as follows;

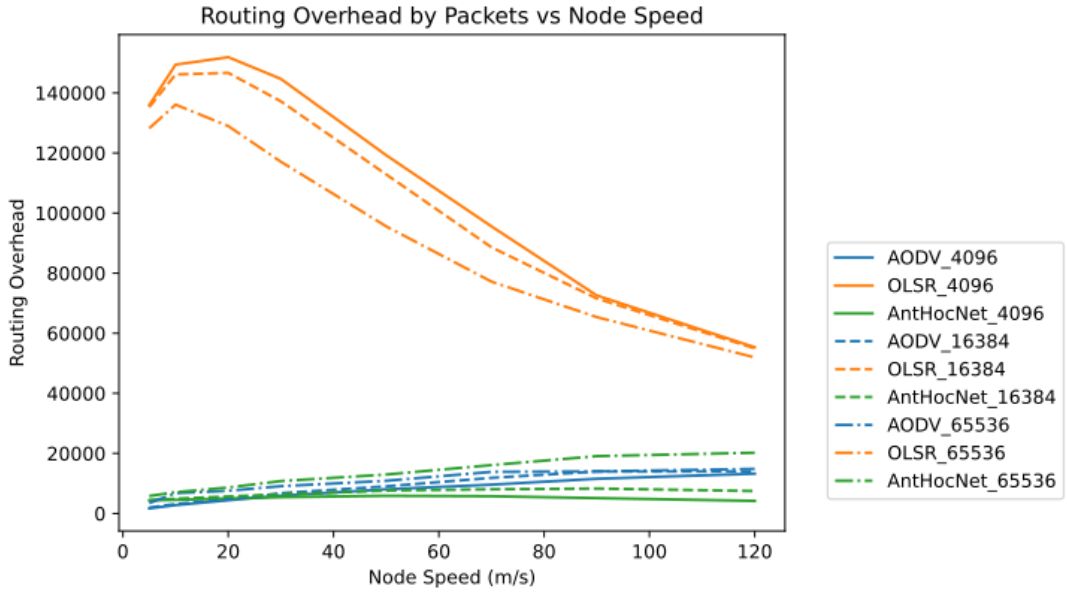


Figure 36: Scenario-II Results: Routing Overhead vs Node Speed (EGM)

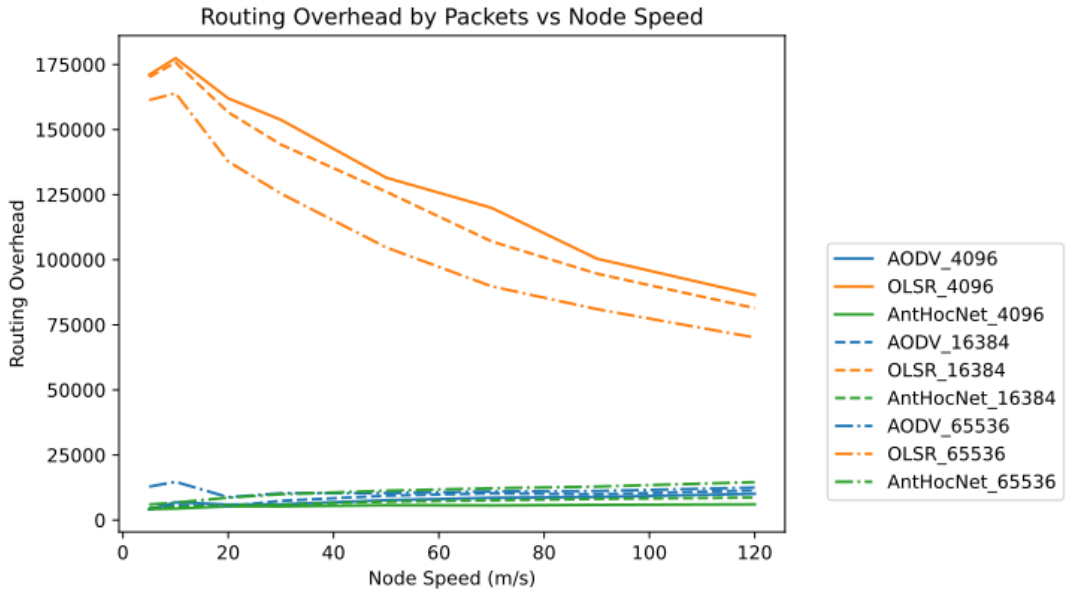


Figure 37: Scenario-II Results: Routing Overhead vs Node Speed (PPRZM)

- Firstly, it can be seen from the results that, for AODV and AntHocNet routing algorithms, as data rate increases, routing overhead also increases in EGM, PPRZM, and RWP mobility models.

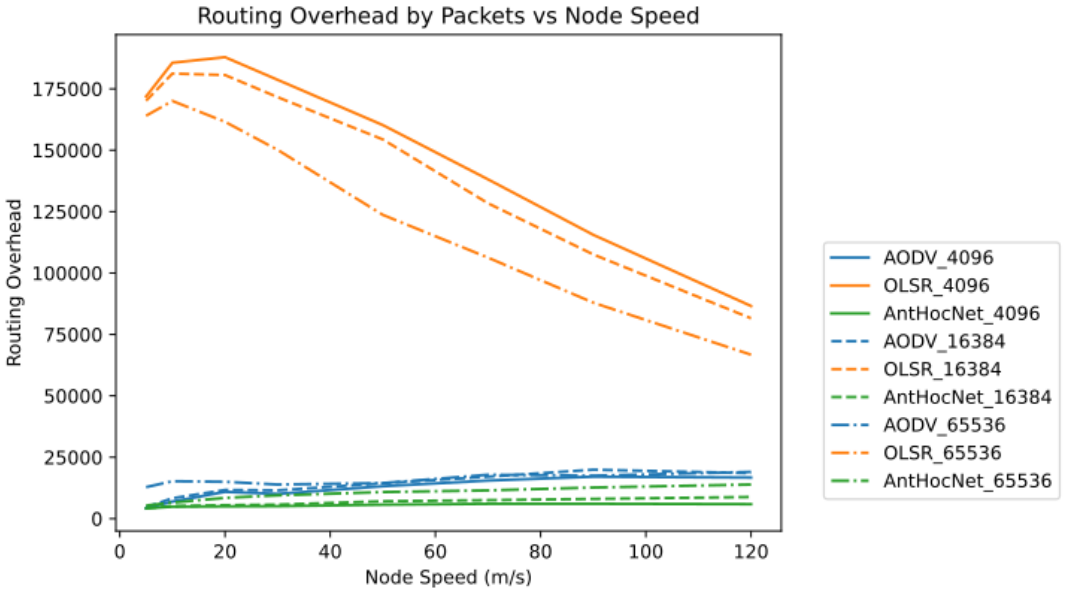


Figure 38: Scenario-II Results: Routing Overhead vs Node Speed (RWP)

It should be noted that the reactive nature of AODV and AntHocNet routing algorithms is the main reason for this situation. When more data packets need to be sent, a route is needed for each transmission. However, as node speed increases, the number of broken links also increases. Therefore, an increase in the route requirement for data transmission demands more routing packets to be sent. In conclusion, both AODV and AntHocNet need to send more routing packets as the data transmission need increases, seen from the presented routing overhead results.

- Secondly, even though AODV and AntHocNet routing algorithms need more routing packets to be sent as data rate increases, routing overhead caused by the OLSR routing algorithm is not expected to be affected by a change in data rate. However, when the results are inspected, the routing overhead of the OLSR routing algorithm decreases as the data rate increases in all of the simulations executed for the selected mobility models.

  This situation can be understood with the packet loss ratio results. As it is stated in Section 5.2.1.1, as the data rate increases, the packet loss rate of the OLSR routing algorithm increases due to network congestion. Following the results analyzed in Section 5.1.1.1, when the packet loss rate increases, the routing overhead for the OLSR routing

algorithm decreases; hence, the dropped periodic routing messages cannot trigger subsequent routing packet broadcasting. In conclusion, an increase in data rate causes the routing overhead of the OLSR routing algorithm to decrease.

The decrease of routing overhead as the node speed increases for the OLSR routing algorithm is analyzed in Section 5.1.1.1.

The results obtained for routing overhead show that the AntHocNet routing algorithm can provide lower routing overhead than AODV even if it contains proactive components for the route maintenance process.

CHAPTER 6

CONCLUSIONS

This research makes three main contributions to the literature. Firstly, it provides a network performance comparison of the bio-inspired routing protocol, namely the AntHocNet, and the traditional routing algorithms, namely the AODV and the OLSR, for FW-UAV ad hoc networks. Secondly, it provides a context where the effect of different mobility models, namely the EGM, PPRZM, and RWP, can be observed on the compared routing algorithms. Lastly, it shows the lower threshold value of the average number of neighbors to provide a non-partitioned network and the effect of the changing node speed on this threshold value.

This chapter presents the significant results concerning these contributions. Also, it mentions the limitations of this research and recommendations for later studies. Finally, concluding remarks are presented.

6.1 Summary

This research provides a basis for future studies by dealing with some of the most critical issues in FW-UAV ad hoc networks.

Firstly, a comparison of various routing protocols from different categories is presented through the NS-3 network simulations. The routing protocols used in this comprehensive study are the AODV, i.e., a reactive routing protocol, OLSR, i.e., a proactive routing protocol, and AntHocNet, i.e., a bio-inspired and hybrid routing protocol. These routing algorithms are tested with scenarios appropriate for FW-FANETs, i.e., high node speeds, 3D simulation space, mobility models offered for FW-UAVs. The comparison of the routing algorithms shows that the

AntHocNet bio-inspired routing algorithm is a viable option for FW-UAV ad hoc networks, i.e., FW-FANETs, and can outperform the traditional routing algorithms, namely the AODV and OLSR.

Secondly, this research provides validation for the appropriate node density, in other words, the average number of neighbors. In the preliminary work of this study, the required value of the average number of neighbors for a non-partitioned network is investigated for the selected routing protocols. Also, the effect of the node speed on this required value of the average number of neighbors is observed. The results show a lower threshold value for the average number of neighbors to provide connectivity, and this threshold value decreases as nodes start to move, i.e., topology starts to change.

Lastly, this research undeniably shows that the mobility model selection affects routing performance, affecting the topology change rate. Therefore, it should be noted that it is essential to select suitable mobility models in mobile or flying, ad hoc network researches.

6.2 Discussion of Significant Results

According to the simulation results, it is clear that the change in topology change rate is one of the most critical factors that affects the performance of the routing protocols. While the dynamicity of the network topology increases, the communication performances of all the routing algorithms deteriorate. However, it is clear that the AntHocNet algorithm, as a hybrid routing protocol, adapts to the increasing dynamicity of the network topology better than traditional routing algorithms, the AODV and the OLSR. In other words, the topology change rate significantly affects route discovery. Even though some of the routing protocols handle an increase in the topology change rate better than others, such as the AntHocNet routing algorithm, an increase in the topology change rate usually damages the route discovery process and degrades the network performance.

Another striking point is the impact of the selected mobility model. It is clear that the selected mobility model directly influences the network performance of a routing protocol, either proactive, reactive, or hybrid. The results indicate that the EGM mobility model increases the topology change rate more than the PPRZM and RWP mobility models as expected; be-

cause the EGM mobility model changes the trajectory dramatically only when the node is close to the border of the simulation space. On the other hand, PPRZM can practically hover if the node is in *Stay-At*, *Eight*, or *Oval* states. Furthermore, RWP can return the neighborhood of the previous position without getting close to the simulation space's border.

Another significant result obtained is that the OLSR routing algorithm is unsuitable for practical use in FW-FANETs. It should be noted that when node speed goes above 20 m/s, the packet loss ratio of the OLSR routing protocol exceeds 0.4 and approaches 1 with increasing node speed. These packet loss ratio values may not be acceptable for practical use cases. On the other hand, AODV and AntHocNet can be used in FW-FANET applications since they can limit the packet loss rate below a reasonable value.

As anticipated for the first research question, the results indicate that the performance of all selected routing algorithms degrade as the topology change rate increases even though the AntHocNet routing algorithm provides better performance than AODV and OLSR.

Another notable result that should be mentioned is concerning increasing data rates. It is evident that the packet transmission interval, in other words, data rate, directly affects the performance of all routing protocols, and consequently, network load has an immediate effect on routing performance. The crucial point in this relationship is that when the topology change rate increases, the effect of the network load on the routing algorithm performances increases. In other words, routing protocols struggle to provide the quality of service, i.e., low packet loss rate, end-to-end delay, and end-to-end jitter, as the topology change rate increases.

As anticipated for the second research question, the results indicate that the performance of all selected routing algorithms degrade as the network load increases, even though the AntHocNet routing algorithm provides better performance than the AODV and the OLSR.

About the factors that affect the connectivity of nodes in FW-UAV ad hoc networks, i.e., concerning the third research question, this study clearly shows two of them.

The first factor that is crucial for a non-partitioned network is the average number of neighbors. In section 4.2.1.1, it is made clear that there is a lower threshold for the value of the average number of neighbors that should be exceeded to obtain a non-partitioned network.

The factor of the average number of neighbors is also presented in [7] for stationary ad hoc networks. The significant contribution of this study is to show that the lower threshold for this factor shifts according to the node speed, i.e., the topology change rate. In other words, as node speed increases, the required value of the average number of neighbors for a non-partitioned network decreases.

The second factor that is significant for connectivity is the selection and the performance of the routing algorithm. The OLSR routing protocol cannot maintain the routes at higher node speeds appropriate for FW-UAVs; hence, the connectivity significantly degrades. It is possible to reduce the TC message interval by changing the implementation parameters of the OLSR routing protocol. By doing this, OLSR may maintain the routes at higher topology change rates. However, this parameter tuning also increases routing overhead significantly. On the other hand, AntHocNet bio-inspired routing algorithm can provide the desired connectivity.

The preliminary simulations conducted to validate the selected value for the average number of neighbors indicate that for the third research question, the anticipated factor, i.e., the average number of neighbors, affects the connectivity of nodes in FW-FANETs.

6.3 Limitations of the Research

The limitations of this research are stated as follows:

1. One of the main limitations of this research is the processing power of the nodes. The results presented in this study heavily depend on the ability of nodes to execute the algorithms.

   This limitation has two aspects. The first aspect is about the ability to execute the algorithm. It is assumed that each node has enough memory and computing power to execute the routing algorithms together with the IP stack and application layer. Secondly, this study assumes that the nodes can execute algorithms in Real-Time and with ignorable processing latency.

2. Limitations of the NS-3 simulator tool apply to this research. The NS-3 simulator tool is a discrete network simulator and does not provide real-time behaviors.

3. The selection of OLSR, AODV, and AntHocNet limits this study. It is possible to extend this research by using different routing algorithms both for the traditional routing algorithms and for the bio-inspired routing algorithm.

4. Another limitation of this study is that the node movements are independent of each other. The results presented in this study do not hold for nodes that have group mobility.

5. This study uses only a selected group of data rates values.

6.4 Recommendations for Future Research

This research provides an inspiring basis for future studies. One of the recommendations for future research is about the effect of the topology change rate on the minimum threshold value of the average number of neighbors for a non-partitioned network. Even though this study clearly shows the effect of the increasing node speed on the minimum threshold value of the average number of neighbors for a non-partitioned network, it is also meaningful to investigate the effect of the mobility models. Another interesting point would be to show how the average number of neighbors affects the routing overhead.

Another aspect that may be investigated is concerning the issue of power consumption. It is possible to extend this study by considering the estimated power consumption of the routing algorithms and comparing the performances of the bio-inspired routing algorithms with energy-aware routing algorithms.

Another future research recommendation is to test bio-inspired algorithms other than AntHocNet. It should be an excellent extension to test a bio-inspired routing algorithm from another group of bio-inspired algorithms, such as the Artificial Bee Colony algorithm from the group of Bee Algorithms. Moreover, it is possible to use traditional routing algorithms other than AODV and OLSR.

Another recommendation may be the use of group mobility models in the simulations. Since group mobility models are also a good fit for specific application areas and a good fit for UAVs, studying the routing performance of the protocols with group mobility models would be a fine addition to this research.

In this research, the nodes communicate with a base node. It is possible to extend this work by making nodes communicate with each other.

Last but not least, it is possible to focus on specific application layers, such as multimedia streaming. Since airborne surveillance is one of the hottest research areas, it would be another excellent extension for this study, and it is crucial to investigate the performance of bio-inspired routing algorithms for such application areas.

REFERENCES

[1] M. Mozaffari, W. Saad, M. Bennis, Y. Nam, and M. Debbah, "A tutorial on uavs for wireless networks: Applications, challenges, and open problems," *IEEE Communications Surveys Tutorials*, vol. 21, no. 3, pp. 2334–2360, 2019.

[2] A. Malhotra and S. Kaur, "A comprehensive review on recent advancements in routing protocols for flying ad hoc networks," *Transactions on Emerging Telecommunications Technologies*, no. June, pp. 1–32, 2019.

[3] Y. Li, J. He, Z. Xie, D. Jiang, and Y. Li, "Frontier development of chips design and production," *Procedia Computer Science*, vol. 139, pp. 554–560, 2018.

[4] H. Zhang, X. Wang, P. Memarmoshrefi, and D. Hogrefe, "A Survey of Ant Colony Optimization Based Routing Protocols for Mobile Ad Hoc Networks," *IEEE Access*, vol. 5, pp. 24139–24161, 2017.

[5] T. Gui, C. Ma, F. Wang, and D. E. Wilkins, "Survey on swarm intelligence based routing protocols for wireless sensor networks: An extensive study," *Proceedings of the IEEE International Conference on Industrial Technology*, vol. 2016-May, pp. 1944–1949, 2016.

[6] A. Nayyar and R. Singh, "Ant Colony Optimization (ACO) based Routing Protocols for Wireless Sensor Networks (WSN): A Survey," *International Journal of Advanced Computer Science and Applications*, vol. 8, no. 2, 2017.

[7] C. Sevgi and A. Koçyiğit, "Optimal deployment in randomly deployed heterogeneous wsns: A connected coverage approach," *Journal of Network and Computer Applications*, vol. 46, pp. 182–197, 2014.

[8] G. Jayakumar and G. Gopinath, "Ad Hoc Mobile Wireless Networks Routing Protocols â€“ A Review," *Journal of Computer Science*, vol. 3, no. 8, pp. 574–582, 2007.

[9] N. Mittal, U. Singh, R. Salgotra, and B. S. Sohi, "A boolean spider monkey optimization based energy efficient clustering approach for WSNs," *Wireless Networks*, vol. 24, no. 6, pp. 2093–2109, 2018.

[10] I. F. Akyildiz, W. Su, Y. Sankarasubramaniam, and E. Cayirci, "Wireless sensor networks: A survey," *Computer Networks*, vol. 38, no. 4, pp. 393–422, 2002.

[11] A. Kout, S. Labed, S. Chikhi, and E. B. Bourennane, "AODVCS, a new bio-inspired routing protocol based on cuckoo search algorithm for mobile ad hoc networks," *Wireless Networks*, vol. 24, no. 7, pp. 2509–2519, 2018.

[12] A. Guillen-Perez and M. D. Cano, "Flying ad hoc networks: A new domain for network communications," *Sensors (Switzerland)*, vol. 18, no. 10, 2018.

[13] M. F. Khan, K. L. A. Yau, R. M. Noor, and M. A. Imran, "Routing schemes in FANETs: A survey," *Sensors (Switzerland)*, vol. 20, no. 1, pp. 1–33, 2020.

[14] M. H. Eiza, T. Owens, Q. Ni, and Q. Shi, "Situation-aware QoS routing algorithm for vehicular ad hoc networks," *IEEE Transactions on Vehicular Technology*, vol. 64, no. 12, pp. 5520–5535, 2015.

[15] H. Nawaz, H. M. Ali, and A. A. Laghari, "Uav communication networks issues: a review," *Archives of Computational Methods in Engineering*, vol. 28, no. 3, pp. 1349–1369, 2021.

[16] I. Bekmezci, O. K. Sahingoz, and a. Temel, "Flying Ad-Hoc Networks (FANETs): A survey," *Ad Hoc Networks*, vol. 11, no. 3, pp. 1254–1270, 2013.

[17] K. Singh and A. K. Verma, "Applying OLSR routing in FANETs," *Proceedings of 2014 IEEE International Conference on Advanced Communication, Control and Computing Technologies, ICACCCT 2014*, no. 978, pp. 1212–1215, 2015.

[18] M. H. Tareque, M. S. Hossain, and M. Atiquzzaman, "On the routing in flying ad hoc networks," *Proceedings of the 2015 Federated Conference on Computer Science and Information Systems, FedCSIS 2015*, vol. 5, pp. 1–9, 2015.

[19] A. Chriki, H. Touati, H. Snoussi, and F. Kamoun, "FANET: Communication, mobility models and security issues," *Computer Networks*, vol. 163, p. 106877, 2019.

[20] I. U. Khan, I. M. Qureshi, M. A. Aziz, T. A. Cheema, and S. B. H. Shah, "Smart IoT control-based nature inspired energy efficient routing protocol for Flying Ad Hoc Network (FANET)," *IEEE Access*, vol. 8, pp. 56371–56378, 2020.

[21] G. Danoy, M. R. Brust, and P. Bouvry, "Connectivity stability in autonomous multi-level UAV swarms for wide area monitoring," *DIVANet 2015 - Proceedings of the 5th ACM Symposium on Development and Analysis of Intelligent Vehicular Networks and Applications*, pp. 1–8, 2015.

[22] R. L. Brouwer, M. A. de Schipper, P. F. Rynne, F. J. Graham, A. J. Reniers, and J. H. MacMahan, "Surfzone monitoring using rotary wing unmanned aerial vehicles," *Journal of Atmospheric and Oceanic Technology*, vol. 32, no. 4, pp. 855–863, 2015.

[23] S. Thounhom and T. Anusas-Amornkul, "The study of routing protocols for UAVs using Paparazzi mobility model with different altitudes," *ACM International Conference Proceeding Series*, pp. 106–111, 2016.

[24] M. Dorigo and T. Stützle, "Ant Colony Optimization: Overview and Recent Advances," *Handbook of Metaheuristics*, pp. 227–263, 2010.

[25] B. Nancharaiah and B. Chandra Mohan, "The performance of a hybrid routing intelligent algorithm in a mobile ad hoc network," *Computers and Electrical Engineering*, vol. 40, no. 4, pp. 1255–1264, 2014.

[26] A. Giagkos and M. S. Wilson, "BeeIP - A Swarm Intelligence based routing for wireless ad hoc networks," *Information Sciences*, vol. 265, pp. 23–35, 2014.

[27] S. Chatterjee and S. Das, "Ant colony optimization based enhanced dynamic source routing algorithm for mobile Ad-hoc network," *Information Sciences*, vol. 295, pp. 67–90, 2015.

[28] A. Nayyar, "Flying adhoc network (fanets): Simulation based performance comparison of routing protocols: Aodv, dsdv, dsr, olsr, aomdv and hwmp," in *2018 International Conference on Advances in Big Data, Computing and Data Communication Systems (icABCD)*, pp. 1–9, Aug 2018.

[29] T. Clausen and P. Jacquet, "Optimized link state routing protocol (olsr)," RFC 3626, RFC Editor, October 2003. `http://www.rfc-editor.org/rfc/rfc3626.txt`.

[30] A. Roy and T. Deb, "Performance Comparison of Routing Protocols in Mobile Ad Hoc Networks," *Lecture Notes in Networks and Systems*, vol. 24, no. 2, pp. 33–48, 2018.

[31] K. Singh and A. K. Verma, "Experimental analysis of AODV, DSDV and OLSR routing protocol for flying adhoc networks (FANETs)," *Proceedings of 2015 IEEE International Conference on Electrical, Computer and Communication Technologies, ICECCT 2015*, pp. 1–4, 2015.

[32] C. Perkins, E. Belding-Royer, and S. Das, "Ad hoc on-demand distance vector (aodv) routing," RFC 3561, RFC Editor, July 2003. `http://www.rfc-editor.org/rfc/rfc3561.txt`.

[33] F. Dressler and O. B. Akan, "A survey on bio-inspired networking," *Computer Networks*, vol. 54, no. 6, pp. 881–900, 2010.

[34] C. Selvaraj, R. Siva Kumar, and M. Karnan, "A survey on application of bio-inspired algorithms," *International Journal of Computer Science and Information Technologies*, vol. 5, no. 1, pp. 366–70, 2014.

[35] S. Bitam, A. Mellouk, and S. Zeadally, "Bio-inspired routing algorithms survey for vehicular ad hoc networks," *IEEE Communications Surveys and Tutorials*, vol. 17, no. 2, pp. 843–867, 2015.

[36] C. Shin and M. Lee, "Swarm-Intelligence-Centric Routing Algorithm for Wireless Sensor Networks," *Sensors (Switzerland)*, vol. 20, no. 18, p. 5164, 2020.

[37] X.-S. Yang and M. Karamanoglu, "Swarm intelligence and bio-inspired computation: An overview," in *Swarm Intelligence and Bio-Inspired Computation* (X.-S. Yang, Z. Cui, R. Xiao, A. H. Gandomi, and M. Karamanoglu, eds.), pp. 3–23, Oxford: Elsevier, 2013.

[38] V. A. Maistrenko, L. V. Alexey, and V. A. Danil, "Experimental estimate of using the ant colony optimization algorithm to solve the routing problem in FANET," *2016 International Siberian Conference on Control and Communications, SIBCON 2016 - Proceedings*, 2016.

[39] A. V. Leonov, "Modeling of bio-inspired algorithms AntHocNet and BeeAdHoc for Flying Ad Hoc Networks (FANETs)," *2016 13th International Scientific-Technical Conference on Actual Problems of Electronic Instrument Engineering, APEIE 2016 - Proceedings*, vol. 2, pp. 90–99, 2016.

[40] F. De Rango, M. Tropea, and P. Fazio, "Bio-inspired routing over FANET in emergency situations to support multimedia traffic," *Proceedings of the International Symposium on Mobile Ad Hoc Networking and Computing (MobiHoc)*, pp. 12–17, 2019.

[41] F. Ducatelle, *Adaptive routing in ad hoc wireless multi-hop networks*. PhD thesis, Inf. Faculty, Della Svizzera Italiana Univ., Lugano, Switzerland, 2007.

[42] S. Buta, S. Baghla, and H. Monga, "Mobility models based performance evaluation of AOMDV routing protocol of MANET," *Internat ional Journal of Applied Research*, vol. 3, no. 1, pp. 82–86, 2017.

[43] M. Appiah, "Performance comparison of mobility models in Mobile Ad Hoc Network (MANET)," *2017 1st International Conference on Next Generation Computing Applications, NextComp 2017*, pp. 47–53, 2017.

[44] J. Harri, F. Filali, and C. Bonnet, "Mobility models for vehicular ad hoc networks: A survey and taxonomy," *IEEE Communications Surveys and Tutorials*, vol. 11, no. 4, pp. 19–41, 2009.

[45] J. Rangaraj and M. Anitha, "Performance analysis of proactive and reactive protocol under different mobility models for manet," in *2017 International Conference on I-SMAC (IoT in Social, Mobile, Analytics and Cloud) (I-SMAC)*, pp. 637–643, 2017.

[46] A. Bujari, C. E. Palazzi, and D. Ronzani, "FANET application scenarios and mobility models," *DroNet 2017 - Proceedings of the 3rd Workshop on Micro Aerial Vehicle Networks, Systems, and Applications, co-located with MobiSys 2017*, pp. 43–46, 2017.

[47] J. D. M. M. Biomo, T. Kunz, and M. St-Hilaire, "An enhanced Gauss-Markov mobility model for simulations of unmanned aerial ad hoc networks," *2014 7th IFIP Wireless and Mobile Networking Conference, WMNC 2014*, 2014.

[48] "Paparazzi." `https://wiki.paparazziuav.org/wiki/Main_Page`. Last Accessed: 2020-10-30.

[49] O. Bouachir, A. Abrassart, F. Garcia, and N. Larrieu, "A mobility model for UAV ad hoc network," *2014 International Conference on Unmanned Aircraft Systems, ICUAS 2014 - Conference Proceedings*, pp. 383–388, 2014.

[50] O. S. Oubbati, M. Atiquzzaman, P. Lorenz, M. H. Tareque, and M. S. Hossain, "Routing in flying Ad Hoc networks: Survey, constraints, and future challenge perspectives," *IEEE Access*, vol. 7, pp. 81057–81105, 2019.

[51] D. Shumeye Lakew, U. Sa'Ad, N. N. Dao, W. Na, and S. Cho, "Routing in Flying Ad Hoc Networks: A Comprehensive Survey," *IEEE Communications Surveys and Tutorials*, vol. 22, no. 2, pp. 1071–1120, 2020.

[52] S. Hayat, E. Yanmaz, and R. Muzaffar, "Survey on Unmanned Aerial Vehicle Networks for Civil Applications: A Communications Viewpoint," *IEEE Communications Surveys and Tutorials*, vol. 18, no. 4, pp. 2624–2661, 2016.

[53] L. Gupta, R. Jain, and G. Vaszkun, "Survey of Important Issues in UAV Communication Networks," *IEEE Communications Surveys and Tutorials*, vol. 18, no. 2, pp. 1123–1152, 2016.

[54] S. Maakar, Y. Singh, and R. Singh, "Performance Investigation of OLSR and AODV Routing Protocols for 3D FANET Environment using NS3 Network Security View project M.Tech View project," *Journal of Communication Engineering and Systems*, vol. 8, no. 2, pp. 1–10, 2018.

[55] A. C. Zucchi and R. M. Silveira, "Performance analysis of routing protocol for ad hoc UAV network," *LANC 2018 - Proceedings of the 10th Latin American Networking Conference*, no. October, pp. 73–80, 2018.

[56] A. V. Leonov, G. A. Litvinov, and D. A. Korneev, "Simulation-based packet delivery performance evaluation with different parameters in flying ad-hoc network (fanet) using olsr," *International Conference of Young Specialists on Micro/Nanotechnologies and Electron Devices, EDM*, vol. 2018-July, pp. 79–85, 2018.

[57] A. Leonov, "Applying Bio-Inspired Algorithms to Routing Problem Solution in FANET," *Bulletin of the South Ural State University. Ser. Computer Technologies, Automatic Control & Radioelectronics*, vol. 17, no. 2, pp. 5–23, 2017.

[58] G. Ravikiran and S. Singh, "Influence of Mobility Models on the performance of routing protocols in Ad-Hoc wireless networks," *IEEE Vehicular Technology Conference*, vol. 59, no. 4, pp. 2185–2189, 2004.

[59] J. D. M. Me Biomo, T. Kunz, M. St-Hilaire, and Y. Zhou, "Unmanned aerial ad hoc networks: Simulation-based evaluation of entity mobility models' impact on routing performance," *Aerospace*, vol. 2, no. 3, pp. 392–422, 2015.

[60] R. Bilal and B. M. Khan, "Analysis of mobility models and routing schemes for flying ad-hoc networks (FANETS)," *International Journal of Applied Engineering Research*, vol. 12, no. 12, pp. 3263–3269, 2017.

[61] J. Hong, D. Zhang, and X. Niu, "Impact Analysis of Node Motion on the Performance of FANET Routing Protocols," *14Th International Conference on Wireless Communications, Networking and Mobile Computing (Wicom 2018)*, vol. 306, no. 118, pp. 147–162, 2018.

[62] H. Ali, S. ul Islam, H. Song, and K. Munir, "A performance-aware routing mechanism for flying ad hoc networks," *Transactions on Emerging Telecommunications Technologies*, vol. 32, no. 1, pp. 1–17, 2021.

[63] S. Maakar, Y. Singh, and R. Singh, "An Enhanced Gauss-Markov Mobility Model for Simulation of FANET in 3-D," *Journal of Advanced Research in Dynamical and Control Systems*, 2018.

[64] A. AlKhatieb, E. Felemban, and A. Naseer, "Performance evaluation of ad-hoc routing protocols in (fanets)," in *2020 IEEE Wireless Communications and Networking Conference Workshops (WCNCW)*, pp. 1–6, April 2020.

[65] "Ns-3 simulator." `https://www.nsnam.org/`. Last Accessed: 2020-11-30.

[66] M. Lacage and T. R. Henderson, "Yet another network simulator," *ACM International Conference Proceeding Series*, vol. 202, 2006.

[67] "Ns-3 olsr-routing-protocol." https://www.nsnam.org/doxygen/classns3_1_1olsr_1_1_routing_protocol.html#details. Last Accessed: 2020-12-06.

[68] "Ns-3 aodv-routing-protocol." https://www.nsnam.org/doxygen/classns3_1_1aodv_1_1_routing_protocol.html#details. Last Accessed: 2020-12-06.

[69] "Anthocnet." http://people.idsia.ch/~frederick/anthocnet/anthocnet.html. Last Accessed: 2020-12-06.

[70] G. Carneiro, P. Fortuna, and M. Ricardo, "Flowmonitor: A network monitoring framework for the network simulator 3 (ns-3)," in *Proceedings of the Fourth International ICST Conference on Performance Evaluation Methodologies and Tools*, VALUETOOLS '09, (Brussels, BEL), ICST (Institute for Computer Sciences, Social-Informatics and Telecommunications Engineering), 2009.

[71] C. M. Demichelis and P. Chimento, "IP Packet Delay Variation Metric for IP Performance Metrics (IPPM)," RFC 3393, RFC Editor, Nov. 2002.

[72] M. A. Boon, A. P. Drijfhout, and S. Tesfamichael, "Comparison of a fixed-wing and multi-rotor UAV for environmental mapping applications: A case study," *International Archives of the Photogrammetry, Remote Sensing and Spatial Information Sciences - ISPRS Archives*, vol. 42, no. 2W6, pp. 47–54, 2017.

APPENDIX A

RESULTS FOR THE AVERAGE NUMBER OF NEIGHBORS

Table 6: PLR vs ANoN for AODV, OLSR and AntHocNet

Node Speed (m/s)	Routing	AvNumofNeighbors	Packet Loss Rate
0	AODV	0.5655	1.0000
0	AODV	1.3404	1.0000
0	AODV	2.6180	0.9173
0	AODV	3.4845	0.4973
0	AODV	3.9816	0.4464
0	AODV	4.5239	0.1844
0	AODV	5.1133	0.1103
0	AODV	5.7517	0.0010
0	AODV	6.4412	0.0010
0	AODV	7.1838	0.0004
0	AODV	8.8357	0.0008
0	OLSR	0.5655	1.0000
0	OLSR	1.3404	1.0000
0	OLSR	2.6180	0.9193
0	OLSR	3.4845	0.5060
0	OLSR	3.9816	0.4535
0	OLSR	4.5239	0.1881
0	OLSR	5.1133	0.1139

Node Speed (m/s)	Routing	AvNumofNeighbors	Packet Loss Rate
0	OLSR	5.7517	0.0000
0	OLSR	6.4412	0.0000
0	OLSR	7.1838	0.0000
0	OLSR	8.8357	0.0000
0	AntHocNet	0.5655	1.0000
0	AntHocNet	1.3404	1.0000
0	AntHocNet	2.6180	0.4272
0	AntHocNet	3.4845	0.2196
0	AntHocNet	3.9816	0.2013
0	AntHocNet	4.5239	0.0797
0	AntHocNet	5.1133	0.0493
0	AntHocNet	5.7517	0.0015
0	AntHocNet	6.4412	0.0014
0	AntHocNet	7.1838	0.0010
0	AntHocNet	8.8357	0.0009
5	AODV	0.5655	1.0000
5	AODV	1.3404	0.6938
5	AODV	2.6180	0.1298
5	AODV	3.4845	0.0519
5	AODV	3.9816	0.0245
5	AODV	4.5239	0.0189
5	AODV	5.1133	0.0128
5	AODV	5.7517	0.0148
5	AODV	6.4412	0.0094
5	AODV	7.1838	0.0129
5	AODV	8.8357	0.0104
5	OLSR	0.5655	1.0000
5	OLSR	1.3404	0.9106
5	OLSR	2.6180	0.2443

Node Speed (m/s)	Routing	AvNumofNeighbors	Packet Loss Rate
5	OLSR	3.4845	0.1902
5	OLSR	3.9816	0.1334
5	OLSR	4.5239	0.1026
5	OLSR	5.1133	0.0757
5	OLSR	5.7517	0.0402
5	OLSR	6.4412	0.0344
5	OLSR	7.1838	0.0426
5	OLSR	8.8357	0.0188
5	AntHocNet	0.5655	1.0000
5	AntHocNet	1.3404	0.3125
5	AntHocNet	2.6180	0.0937
5	AntHocNet	3.4845	0.0375
5	AntHocNet	3.9816	0.0320
5	AntHocNet	4.5239	0.0219
5	AntHocNet	5.1133	0.0123
5	AntHocNet	5.7517	0.0107
5	AntHocNet	6.4412	0.0091
5	AntHocNet	7.1838	0.0126
5	AntHocNet	8.8357	0.0043

Table 7: PLR vs ANoN for AntHocNet with Various Speeds

Node Speed (m/s)	AvNumofNeighbors	Packet Loss Rate
0	0.56549	1.00000
0	1.34041	1.00000
0	2.61799	0.34621
0	3.48455	0.20114
0	3.98164	0.13935
0	4.52389	0.06163

Node Speed (m/s)	AvNumofNeighbors	Packet Loss Rate
0	5.11327	0.04969
0	5.75173	0.02530
0	6.44125	0.00138
0	7.18378	0.00105
0	10.72330	0.00087
1	0.56549	1.00000
1	1.34041	0.74246
1	2.61799	0.22300
1	3.48455	0.12014
1	3.98164	0.07784
1	4.52389	0.05243
1	5.11327	0.03830
1	5.75173	0.01610
1	6.44125	0.00476
1	7.18378	0.00173
1	10.72330	0.00113
5	0.56549	1.00000
5	1.34041	0.32858
5	2.61799	0.10136
5	3.48455	0.03894
5	3.98164	0.03467
5	4.52389	0.02493
5	5.11327	0.01567
5	5.75173	0.01318
5	6.44125	0.01219
5	7.18378	0.00999
5	10.72330	0.00544
10	0.56549	0.48495
10	1.34041	0.32617

Node Speed (m/s)	AvNumofNeighbors	Packet Loss Rate
10	2.61799	0.09683
10	3.48455	0.07086
10	3.98164	0.06630
10	4.52389	0.06489
10	5.11327	0.05423
10	5.75173	0.04779
10	6.44125	0.06247
10	7.18378	0.04469
10	10.72330	0.02089

APPENDIX B

MOBILITY MODEL USER CODES

```
MobilityHelper mobility helper;

int64 t streamIndex = 0; // used to get consistent mobility across scenarios

ObjectFactory pos;
pos.SetTypeId ("ns3::RandomBoxPositionAllocator");
pos.Set ("X", StringValue ("ns3::UniformRandomVariable[Min=0.0 Max=2000.0]"));
pos.Set ("Y", StringValue ("ns3::UniformRandomVariable[Min=0.0 Max=2000.0]"));
pos.Set ("Z", StringValue ("ns3::UniformRandomVariable[Min=0.0 Max=1500.0]"));

Ptr PositionAllocator taPositionAlloc = pos.Create () GetObject PositionAllocator ();
streamIndex += taPositionAlloc AssignStreams (streamIndex);

std::stringstream ssSpeed;
ssSpeed "ns3::UniformRandomVariable[Min=" (p config nodeSpeed 5) "Max=" (p config nodeSpeed + 5)
        "]";

mobility helper.SetMobilityModel (
"ns3::EnhancedGaussMarkovMobilityModel",
"Bounds", BoxValue (Box (0, 2000, 0, 2000, 0, 1500)),
"TimeStep", TimeValue (Seconds (0.5)),
"Alpha", DoubleValue (0.85),
"MeanVelocity", StringValue (ssSpeed.str()),
"MeanDirection", StringValue ("ns3::UniformRandomVariable[Min=0 Max=6.283185307]"),
"MeanPitch", StringValue ("ns3::UniformRandomVariable[Min=0.05 Max=0.05]"),
"NormalVelocity", StringValue ("ns3::NormalRandomVariable[Mean=0.0 Variance=0.0 Bound=0.0]"),
"NormalDirection", StringValue ("ns3::NormalRandomVariable[Mean=0.0 Variance=0.2 Bound=0.4]"),
"NormalPitch", StringValue ("ns3::NormalRandomVariable[Mean=0.0 Variance=0.02 Bound=0.04]")
);

mobility helper.SetPositionAllocator (taPositionAlloc);
mobility helper.Install(nodes);

std::cout "EGM mobility Selected";
```

Listing B.1: EGM User Code

```
MobilityHelper mobility helper;

int64 t streamIndex = 0; // used to get consistent mobility across scenarios

ObjectFactory pos;
pos.SetTypeId ("ns3::RandomBoxPositionAllocator");
```

```
6  pos.Set ("X", StringValue ("ns3::UniformRandomVariable[Min=0.0 Max=2000.0]"));
7  pos.Set ("Y", StringValue ("ns3::UniformRandomVariable[Min=0.0 Max=2000.0]"));
8  pos.Set ("Z", StringValue ("ns3::UniformRandomVariable[Min=0.0 Max=1500.0]"));
9
10 Ptr PositionAllocator taPositionAlloc = pos.Create () GetObject PositionAllocator ();
11 streamIndex += taPositionAlloc AssignStreams (streamIndex);
12
13 std::stringstream ssSpeed;
14 ssSpeed pconfig nodeSpeed;
15
16 mobility helper.SetMobilityModel (
17   "ns3::PaparazziMobilityModel",
18   "Bounds", BoxValue (Box (0, 2000, 0, 2000, 0, 1500)),
19   "Speed", StringValue (ssSpeed.str ()),
20   "TurnRadius", DoubleValue(30),
21   "PositionAllocator", PointerValue (taPositionAlloc));
22
23 mobility helper.SetPositionAllocator (taPositionAlloc);
24 mobility helper.Install(nodes);
25
26 std::cout "Paparazzi mobility Selected";
```

Listing B.2: PPRZM User Code

APPENDIX C

DATA COLLECTOR IMPLEMENTATIONS

```
/
   flow monitor stats reader.cc

    Created on: Mar 23, 2020
        Author: burak seker
 /

include "flow monitor stats reader.h"

include  boost/filesystem/path.hpp
include  sys/stat.h
include  fstream
include  ios
include  boost/filesystem.hpp
include  algorithm
include  tuple
include  vector

include  ns3/gnuplot.h
include  ns3/histogram.h

using std:: codecvt abstract base;
using std::get;
using std::ios;
using std::ofstream;
using std::tuple;
using std::vector;

include  iostream
namespace ns3

using ns3::FlowMonitor;

define UDPPACKETHEADERSIZE  28

typedef vector  tuple  double,double    tuple list;

void FlowMonitorStatsReader::ReadFlowMonitorStats(Ptr FlowMonitor  flowmon,  uint32t packetsize,
    SOLSRMSGSINFO  polsrmsgsinfo, uint32t runnum)
  this  pFlowMonitor = flowmon;

  flowmon  CheckForLostPackets();
  const FlowMonitor::FlowStatsContainer  flowStats = pFlowMonitor  GetFlowStats();

```

```
44  uint64_t total_rx_bytes = 0;
45  uint64_t total_tx_bytes = 0;
46  uint32_t total_rx_packets = 0;
47  uint32_t total_tx_packets = 0;
48  uint32_t total_lost_packets = 0;
49  Time total_delay = MilliSeconds(0);
50  Time total_jitter = MilliSeconds(0);
51  uint32_t total_forwarded_packet_num = 0;
52  uint32_t total_rx_data_packets = 0;
53
54
55  Time total_rx_delta = MilliSeconds(0);
56  Time total_tx_delta = MilliSeconds(0);
57
58  uint32_t total_flow_count = 0;
59  tuple_list delay_hist_data;
60
61  for (FlowMonitor::FlowStatsContainerCI flowI = flowStats.begin (); flowI != flowStats.end (); flowI++)
62    total_flow_count++;
63    total_rx_bytes += flowI->second.rxBytes;
64    total_tx_bytes += flowI->second.txBytes;
65    total_rx_packets += flowI->second.rxPackets;
66    total_tx_packets += flowI->second.txPackets;
67
68    total_tx_delta += flowI->second.timeLastTxPacket - flowI->second.timeFirstTxPacket;
69
70    if(flowI->second.packetSizeHistogram.GetNBins() == packet_size + UDPPACKETHEADERSIZE + 1 )
71      //flow is dataPacketFlow
72      uint32_t data_packet_cnt = flowI->second.packetSizeHistogram.GetBinCount(packet_size +
       UDPPACKETHEADERSIZE);
73      if(data_packet_cnt > 0)
74        total_rx_data_packets += data_packet_cnt;
75
76        total_lost_packets += flowI->second.lostPackets;
77        total_delay += flowI->second.delaySum;
78        total_jitter += flowI->second.jitterSum;
79        total_forwarded_packet_num += flowI->second.timesForwarded;
80        total_rx_delta += flowI->second.timeLastRxPacket - flowI->second.timeFirstRxPacket;
81
82        Histogram hist = flowI->second.delayHistogram;
83        uint128_t bins=flowI->second.delayHistogram.GetNBins();
84        for (uint128_t i=0;i<bins;i++)
85          delay_hist_data.push_back(tuple<double,double> (hist.GetBinStart(i),hist.GetBinCount(i)));
86
87
88      else
89      total_rx_data_packets ++;
90      total_rx_data_packets --;
91
92
93
94  // Instantiate the dataset, set its title, and make the points be
95  // plotted along with connecting lines.
96  std::string fileNameWithNoExtension = "output_data/delay_hist_" + std::to_string(runnum);
97  std::string graphicsFileName = "output_data/delay_hist_" + std::to_string(runnum) + ".png";
98  std::string plotFileName = "output_data/delay_hist_" + std::to_string(runnum) + ".plt";
99  std::string plotTitle = "Delay_Histogram";
100 std::string dataTitle = "Mean_Delay";
101
102 // Instantiate the plot and set its title.
103 Gnuplot plot (graphicsFileName);
```

```
104   plot.SetTitle (plotTitle);
105   plot.SetOutputFilename("delay hist " + std::to string(runnum) + ".png");
106
107   // Make the graphics file, which the plot file will create when it
108   // is used with Gnuplot, be a PNG file.
109   plot.SetTerminal ("png");
110
111   // Set the labels for each axis.
112   plot.SetLegend ("Mean Delay", "Occurence");
113
114   Gnuplot2dDataset dataset;
115   dataset.SetTitle (dataTitle);
116   dataset.SetStyle (Gnuplot2dDataset::LINESPOINTS);
117
118   sort(delay hist data.begin(), delay hist data.end());
119
120   double val = 0;
121
122   for(uint32 t i=0; i    delay hist data.size(); i++)
123     if(i == 0)
124       val = get 1 (delay hist data[i]);
125       else
126       if(fabs(get 0 (delay hist data[i])    get 0 (delay hist data[i    1]))  1e 9 )
127         val +=  get 1 (delay hist data[i]);
128         else
129         dataset.Add(get 0 (delay hist data[i 1]),val);
130         val = get 1 (delay hist data[i]);
131
132
133       if(i == (delay hist data.size()    1))
134         //last tuple
135         dataset.Add(get 0 (delay hist data[i]),val);
136
137
138
139
140   // Add the dataset to the plot.
141   plot.AddDataset (dataset);
142
143   // Open the plot file.
144   std::ofstream plotFile (plotFileName.c str());
145
146   // Write the plot file.
147   plot.GenerateOutput (plotFile);
148
149   // Close the plot file.
150   plotFile.close ();
151
152   if(0 == total flow count     0 == total rx data packets)
153     meanDelayMs = 0;
154     meanJitterMs = 0;
155     meanHopCount = 0;
156     meanRxBitRateBps = 0;
157     meanRxPacketSizeByte = 0;
158
159     packetLoss = 1;
160     packetLossWithControlMsgs = 1;
161
162     meanTxPacketSizeByte = 0;
163     meanTxBitRateBps = 0;
164     routingOverheadPackets = 1;
```

```
165       routingOverheadBytes = 1;
166
167       else
168       meanDelayMs = totalDelay.GetMilliSeconds() / (float)totalRxDataPackets;
169       meanJitterMs = totalJitter.GetMilliSeconds() / (float)(totalRxDataPackets);
170       meanHopCount = 1+ totalForwardedPacketNum /(float)totalRxDataPackets;
171       meanRxBitRateBps = (8    totalRxDataPackets    (packetSize + UDPPACKETHEADERSIZE)) / (float)
            totalRxDelta.GetSeconds();
172       meanRxPacketSizeByte = totalRxBytes / (float)totalRxPackets;
173
174       packetLoss = totalLostPackets / (float)(totalLostPackets + totalRxDataPackets);
175
176       meanTxPacketSizeByte = totalTxBytes / (float)totalTxPackets;
177       meanTxBitRateBps = (8    totalTxBytes) / (float)totalTxDelta.GetSeconds();
178
179       //means olsr packages. Flowmon cannot filter olsr control messages. So take it from outside
180       if(polsrmsgsinfo    rxpacketcount    0)
181         routingOverheadPackets = polsrmsgsinfo    rxpacketcount;
182         routingOverheadBytes = polsrmsgsinfo    rxbytecount;
183
184         packetLossWithControlMsgs = (totaltxpackets    totalrxpackets)/ (float)totaltxpackets;
185
186         else
187
188         packetLossWithControlMsgs = (totaltxpackets    totalrxpackets)/ (float)totaltxpackets;
189
190         routingOverheadPackets = totalrxpackets    totalrxdatapackets;
191         routingOverheadBytes = totalrxbytes    (totalrxdatapackets    (packetsize + UDPPACKETHEADERSIZE));
192
193
194
195
196 void FlowMonitorStatsReader::SaveStats(string path, string filename)
197
198   boost::filesystem::path dir(path);
199   boost::filesystem::create directory(dir);
200
201   ofstream file;
202
203   file.open(path+boost::filesystem::path::separator +filename, ios::out);
204
205   file    "meanDelayMs"    "t"
206         "meanJitterMs"    "t"
207         "packetLoss"    "t"
208         "packetLossWithControlMsgs"    "t"
209         "meanRxPacketSizeByte"    "t"
210         "meanTxBitRateBps"    "t"
211         "meanRxBitRateBps"    "t"
212         "meanHopCount"    "t"
213         "routingOverheadPackets"    "t"
214         "routingOverheadBytes"    "t"
215         std::endl;
216
217   file    meanDelayMs    "t"
218       meanJitterMs    "t"
219       packetLoss    "t"
220       packetLossWithControlMsgs    "t"
221       meanRxPacketSizeByte    "t"
222       meanTxBitRateBps    "t"
223       meanRxBitRateBps    "t"
224       meanHopCount    "t"
```

```
225         routingOverheadPackets     " t"
226         routingOverheadBytes     " t"
227         std::endl;
228     file.close();
229
230
231
232 void FlowMonitorStatsReader::SerializeHistogram(std::string name, Histogram hist)
233
234     std::stringstream fname;
235     std::stringstream fpname;
236     fpname     name     ".gnuplot";
237     fname     name     ".jpg";
238
239     Gnuplot plot (fname.str());
240     plot.SetTitle("Delay Histogram");
241     plot.SetTerminal("jpeg");
242     plot.SetLegend("Delay (ms)","probability");
243     Gnuplot2dDataset ds;
244     ds.SetTitle("Delay");
245     ds.SetStyle(Gnuplot2dDataset::HISTEPS);
246     uint128t bins=hist.GetNBins();
247     for (uint128t i=0;i bins;i++)
248       ds.Add(hist.GetBinStart(i),hist.GetBinCount(i));
249
250     plot.AddDataset(ds);
251     std::ofstream plotFile (fpname.str().c str());
252     plot.GenerateOutput(plotFile);
253     plotFile.close();
254
255
256
257
```

Listing C.1: FlowMonitorStatsReader Implementation

```
0  import os
1  from os import path
2  import shutil
3  from datetime import datetime
4  import statistics
5  from typing import List, Union
6  import numpy as np
7  from random import seed
8  from random import randint
9
10 from numpy import number
11
12 import TestConfig
13
14 run loop range stop = TestConfig.total run num + 1
15
16 delay dic = dict()
17 delay err dic = dict()
18
19 jitter dic = dict()
20 pkt loss dic = dict()
21 tx pkt size dic = dict()
22 rx pkt size dic = dict()
23 tx bps dic = dict()
```

```
rx_bps_dic = dict()
mean_hop_dic = dict()
routing_overhead_by_packet_dic = dict()
routing_overhead_by_byte_dic = dict()

now = datetime.now()
date_time_str = now.strftime("%d_%m_%Y_%H_%M_%S")


def run_test(num_of_node, router, node_speed_ms, traffic_cbr_bps, test_name, mobility):
    global delay_dic, date_time_str

    test_out_main_dir = TestConfig.test_out_main_dir
    test_out_main_dir = test_out_main_dir + "/" + test_name + "/" + date_time_str

    # Run NS3 Tests
    os.chdir(TestConfig.ns3_dir)
    cmd_str = "./waf run \"routing_simulator "
              " runNumber=%run:d seed=%seed:d "
              " nodeNumber=%nodeNum:d "
              " routingType=%routing:d "
              " time=%simTime:d "
              " dataRateBps=%dataRate:d "
              " nodeSpeed=%nodeSpeed:d "
              " mobilityType=%mobilityType:d \""
    for x in range(1, run_loop_range_stop, 1):
        seed(x)
        os.system(cmd_str.format(run=x, seed=1234, nodeNum=num_of_node, routing=router, simTime=120,
                                 dataRate=traffic_cbr_bps, nodeSpeed=node_speed_ms, mobilityType=mobility))

    # Create test folder with datetime
    os.chdir(TestConfig.ns3_repo_dir)
    test_out_cur_dir = "%0/Node_%1_R_%2_M_%3_DataRate_%4_NSpeed_%5".format(test_out_main_dir, str(
        num_of_node),
                                                                           str(router), str(mobility),
                                                                           str(traffic_cbr_bps),
                                                                           str(node_speed_ms))

    if not os.path.exists(test_out_cur_dir):
        os.makedirs(test_out_cur_dir)

    # move all output files
    files = os.listdir(TestConfig.ns3_out_dir)
    for f in files:
        shutil.move(TestConfig.ns3_out_dir + "/" + f, test_out_cur_dir)
    # Aggregate results
    os.chdir(test_out_cur_dir)
    file_name_base = "flow_reader_stats"

    agg_res_names: List[str] = []
    agg_results: List[Union[List[float]]] = []

    num_of_failed_runs = 0

    for x in range(1, run_loop_range_stop, 1):
        if path.exists(file_name_base + str(x)):
            file = open(file_name_base + str(x), 'r')
            lines = file.readlines()
            results = [float(i) for i in lines[1].strip().split('\t')]

            if x == 1:
```

```
84                  aggresnames = lines[0].strip().split('\t')
85
86              if 1 != results[TestConfig.indpacketLoss]:
87                  aggresults.append(results)
88
89              file.close()
90          else:
91              numoffailedruns = numoffailedruns + 1
92              print("Num OF Failed Runs: %d" % numoffailedruns)
93
94      arrt = np.array(aggresults).T
95      means = np.zeros((TestConfig.indloop range stop, 1))
96      medians = np.zeros((TestConfig.indloop range stop, 1))
97      std = np.zeros((TestConfig.indloop range stop, 1))
98      for x in range(0, TestConfig.indloop range stop):
99          means[x] = statistics.mean(arrt[x])
100         medians[x] = statistics.median(arrt[x])
101         std[x] = statistics.stdev(arrt[x], xbar=means[x][0])
102
103     with open("results mean median std", "w+") as outfile:
104         outfile.write("\t".join(str(item) for item in means))
105         outfile.write("\n")
106         outfile.write("\t".join(str(item) for item in medians))
107         outfile.write("\n")
108         outfile.write("\t".join(str(item) for item in std))
109
110     delaydic[TestConfig.variableparameterval] = means[TestConfig.indmeanDelayMs]
111     delayerrdic[TestConfig.variableparameterval] = std[TestConfig.indmeanDelayMs]
112
113     jitterdic[TestConfig.variableparameterval] = means[TestConfig.indmeanJitterMs]
114     pktlossdic[TestConfig.variableparameterval] = means[TestConfig.indpacketLoss]
115     txpktsizedic[TestConfig.variableparameterval] = means[TestConfig.indmeanTxPacketSizeByte]
116     rxpktsizedic[TestConfig.variableparameterval] = means[TestConfig.indmeanRxPacketSizeByte]
117     txbpsdic[TestConfig.variableparameterval] = means[TestConfig.indmeanTxBitRateBps]
118     rxbpsdic[TestConfig.variableparameterval] = means[TestConfig.indmeanRxBitRateBps]
119     meanhopdic[TestConfig.variableparameterval] = means[TestConfig.indmeanHopCount]
120     routingoverheadbypacketdic[TestConfig.variableparameterval] = means[TestConfig.
          indroutingOverheadPackets]
121     routingoverheadbybytedic[TestConfig.variableparameterval] = means[TestConfig.indroutingOverheadBytes]
122
123     os.chdir(testoutmaindir)
124     print("Num OF Failed Runs: %d" % numoffailedruns)
```

Listing C.2: TestRunner Implementation In Python

APPENDIX D

SIMULATION RESULTS

The tables provided in Appendix D are using the following symbology:

- R : Routing Type
- M : Mobility Type
- DR : Data Rate (bps)
- NS : Node Speed(m/s)
- D : End-to-End Delay (ms)
- HC : Mean Hop Count
- J : Jitter (ms)
- PLR : Packet Loss Ratio
- RO : Routing Overhead

D.1 Scenario1 Data

Table 8: Scenario1 Data

R	M	NS	D	HC	J	PLR	RO

R	M	NS	D	HC	J	PLR	RO
AODV	EGM	5	6.491	3.291	4.849	0.021	1500
AODV	EGM	10	7.261	3.022	7.069	0.032	2436
AODV	EGM	20	13.073	3.020	13.020	0.073	4965
AODV	EGM	30	20.678	3.239	17.987	0.124	6441
AODV	EGM	50	25.573	3.308	25.651	0.189	8925
AODV	EGM	70	27.770	3.410	24.297	0.240	11706
AODV	EGM	90	39.678	3.496	39.519	0.277	12348
AODV	EGM	120	34.930	3.455	21.154	0.298	13140
OLSR	EGM	5	4.320	2.768	2.632	0.140	134463
OLSR	EGM	10	7.675	2.729	5.373	0.220	149103
OLSR	EGM	20	13.121	2.788	9.035	0.448	149009
OLSR	EGM	30	17.762	2.773	12.849	0.584	139383
OLSR	EGM	50	14.039	2.669	8.463	0.806	118524
OLSR	EGM	70	9.959	2.597	2.996	0.950	93196
OLSR	EGM	90	7.878	2.249	2.273	0.984	71889
OLSR	EGM	120	7.318	2.000	1.338	0.996	55246
AntHocNet	EGM	5	2.680	3.915	1.273	0.027	4269
AntHocNet	EGM	10	3.069	3.569	1.603	0.053	4619
AntHocNet	EGM	20	5.842	3.669	4.106	0.071	5278
AntHocNet	EGM	30	10.215	3.998	7.365	0.104	6148
AntHocNet	EGM	50	14.482	3.926	13.368	0.133	6742
AntHocNet	EGM	70	19.206	3.926	19.459	0.190	6744
AntHocNet	EGM	90	19.266	4.073	19.203	0.259	6505
AntHocNet	EGM	120	22.319	3.933	24.771	0.340	4985
AODV	PPRZM	5	7.724	2.626	6.894	0.035	3789
AODV	PPRZM	10	12.719	2.608	11.821	0.056	7945
AODV	PPRZM	20	14.459	2.794	12.729	0.071	5332
AODV	PPRZM	30	13.302	2.744	13.299	0.099	6879
AODV	PPRZM	50	24.721	2.831	20.760	0.152	8185

R	M	NS	D	HC	J	PLR	RO
AODV	PPRZM	70	36.933	2.658	24.175	0.205	9294
AODV	PPRZM	90	46.256	2.578	26.843	0.223	9573
AODV	PPRZM	120	57.145	2.542	35.094	0.245	10647
OLSR	PPRZM	5	5.983	2.380	3.797	0.066	170474
OLSR	PPRZM	10	8.334	2.296	6.169	0.181	179680
OLSR	PPRZM	20	9.261	2.374	7.032	0.354	159358
OLSR	PPRZM	30	14.893	2.278	11.563	0.476	150392
OLSR	PPRZM	50	24.468	2.302	20.384	0.656	132998
OLSR	PPRZM	70	38.548	2.376	36.892	0.746	115643
OLSR	PPRZM	90	37.484	2.052	28.803	0.775	99727
OLSR	PPRZM	120	64.814	2.071	52.013	0.833	86169
AntHocNet	PPRZM	5	2.401	3.356	1.247	0.018	4246
AntHocNet	PPRZM	10	4.791	3.196	3.438	0.056	4792
AntHocNet	PPRZM	20	7.058	3.501	6.232	0.074	5073
AntHocNet	PPRZM	30	8.640	3.539	6.567	0.108	5613
AntHocNet	PPRZM	50	13.275	3.527	10.372	0.108	6120
AntHocNet	PPRZM	70	16.047	3.437	15.091	0.145	6225
AntHocNet	PPRZM	90	16.942	3.405	15.968	0.173	6856
AntHocNet	PPRZM	120	22.776	3.295	21.572	0.209	7239
AODV	RWP	5	6.770	2.739	6.438	0.037	3731
AODV	RWP	10	14.385	2.485	13.994	0.070	7752
AODV	RWP	20	15.284	2.508	14.339	0.082	7033
AODV	RWP	30	19.888	2.566	19.501	0.103	10483
AODV	RWP	50	24.895	2.522	25.523	0.141	13576
AODV	RWP	70	28.123	2.575	30.882	0.168	15157
AODV	RWP	90	50.380	2.505	38.245	0.202	17121
AODV	RWP	120	59.995	2.383	38.032	0.234	17699
OLSR	RWP	5	7.233	2.377	4.845	0.056	170071
OLSR	RWP	10	8.484	2.191	6.170	0.159	185647

R	M	NS	D	HC	J	PLR	RO
OLSR	RWP	20	12.313	2.226	8.869	0.309	184571
OLSR	RWP	30	19.966	2.276	16.993	0.464	176490
OLSR	RWP	50	22.360	2.192	17.135	0.604	160487
OLSR	RWP	70	27.411	2.035	20.524	0.719	134937
OLSR	RWP	90	23.136	1.941	19.190	0.822	113212
OLSR	RWP	120	42.385	1.885	32.106	0.898	86893
AntHocNet	RWP	5	2.561	3.231	0.959	0.011	4280
AntHocNet	RWP	10	3.641	3.205	2.508	0.064	4388
AntHocNet	RWP	20	7.687	3.199	6.120	0.079	4851
AntHocNet	RWP	30	11.061	3.291	8.897	0.094	5296
AntHocNet	RWP	50	15.146	3.317	15.389	0.129	5890
AntHocNet	RWP	70	19.381	3.061	18.714	0.163	6718
AntHocNet	RWP	90	19.949	3.027	20.331	0.190	6601
AntHocNet	RWP	120	22.250	2.958	25.602	0.225	6597

D.2 Scenario2 Data

D.2.1 Simulation Data for EGM Mobility Model

Table 9: Scenario2 Data for EGM Mobility Model

R	DR	NS	D	HC	J	PLR	RO
AODV	4096 bps	5	8.528	3.118	6.921	0.020	1641
AODV	4096 bps	10	11.718	3.063	12.435	0.032	2521
AODV	4096 bps	20	21.086	3.065	21.031	0.066	4409
AODV	4096 bps	30	27.749	3.496	31.489	0.083	6390
AODV	4096 bps	50	37.174	3.437	37.207	0.162	8194
AODV	4096 bps	70	34.585	3.442	42.949	0.203	10709
AODV	4096 bps	90	78.336	3.571	69.016	0.247	10925

R	DR	NS	D	HC	J	PLR	RO
AODV	4096 bps	120	46.358	3.406	35.305	0.270	11548
OLSR	4096 bps	5	5.492	2.761	3.205	0.121	134875
OLSR	4096 bps	10	8.511	2.691	6.320	0.223	149288
OLSR	4096 bps	20	13.383	2.738	10.291	0.441	151078
OLSR	4096 bps	30	16.675	2.726	14.419	0.590	142988
OLSR	4096 bps	50	20.950	2.671	17.502	0.795	120381
OLSR	4096 bps	70	12.221	2.716	4.196	0.946	93933
OLSR	4096 bps	90	10.161	2.207	3.438	0.984	75637
OLSR	4096 bps	120	8.083	2.000	2.087	0.997	55757
AntHocNet	4096 bps	5	3.890	3.714	2.707	0.027	4218
AntHocNet	4096 bps	10	4.969	3.661	3.586	0.038	4662
AntHocNet	4096 bps	20	8.786	3.823	7.642	0.063	5077
AntHocNet	4096 bps	30	13.105	3.884	12.549	0.084	5492
AntHocNet	4096 bps	50	17.346	3.785	18.642	0.118	5770
AntHocNet	4096 bps	70	23.423	3.802	30.240	0.174	5865
AntHocNet	4096 bps	90	25.661	3.954	32.963	0.243	5302
AntHocNet	4096 bps	120	29.525	3.859	38.605	0.312	4356
AODV	16384 bps	5	5.090	3.114	3.496	0.026	1778
AODV	16384 bps	10	6.178	3.030	4.480	0.032	2687
AODV	16384 bps	20	9.832	3.040	8.615	0.100	5212
AODV	16384 bps	30	14.955	3.349	11.063	0.147	7109
AODV	16384 bps	50	18.499	3.267	14.627	0.228	9334
AODV	16384 bps	70	21.802	3.494	17.691	0.271	13220
AODV	16384 bps	90	25.385	3.282	18.750	0.351	12563
AODV	16384 bps	120	23.660	3.558	16.030	0.379	12968
OLSR	16384 bps	5	3.705	2.761	2.472	0.149	133664
OLSR	16384 bps	10	6.520	2.766	4.570	0.232	147920
OLSR	16384 bps	20	11.380	2.817	8.098	0.457	145787
OLSR	16384 bps	30	17.178	2.749	12.763	0.590	134370

R	DR	NS	D	HC	J	PLR	RO
OLSR	16384 bps	50	13.723	2.730	10.947	0.825	115963
OLSR	16384 bps	70	8.956	2.468	3.360	0.957	89597
OLSR	16384 bps	90	6.005	2.201	1.799	0.984	71821
OLSR	16384 bps	120	5.673	2.000	1.158	0.996	54935
AntHocNet	16384 bps	5	2.182	3.589	0.735	0.027	4541
AntHocNet	16384 bps	10	3.533	3.620	1.787	0.075	4999
AntHocNet	16384 bps	20	5.590	3.734	3.295	0.094	5624
AntHocNet	16384 bps	30	8.528	3.996	5.197	0.107	6556
AntHocNet	16384 bps	50	10.100	3.811	6.241	0.156	7653
AntHocNet	16384 bps	70	14.420	3.931	10.216	0.214	8619
AntHocNet	16384 bps	90	15.006	4.178	10.925	0.285	8753
AntHocNet	16384 bps	120	17.391	4.100	12.565	0.366	7400
AODV	65536 bps	5	24.280	3.133	6.947	0.084	5299
AODV	65536 bps	10	16.150	3.003	4.669	0.138	5601
AODV	65536 bps	20	23.583	3.038	7.965	0.216	8180
AODV	65536 bps	30	19.030	3.197	6.799	0.296	9373
AODV	65536 bps	50	28.541	3.044	9.206	0.427	11840
AODV	65536 bps	70	23.635	3.270	10.474	0.507	13758
AODV	65536 bps	90	24.534	3.189	10.113	0.564	14238
AODV	65536 bps	120	27.315	3.475	9.908	0.600	14404
OLSR	65536 bps	5	17.584	2.878	6.530	0.183	128372
OLSR	65536 bps	10	24.696	2.762	8.130	0.266	134045
OLSR	65536 bps	20	23.926	2.818	11.729	0.509	128909
OLSR	65536 bps	30	22.722	2.785	11.323	0.652	113518
OLSR	65536 bps	50	16.103	2.626	8.780	0.859	94971
OLSR	65536 bps	70	19.553	2.393	8.658	0.959	78411
OLSR	65536 bps	90	9.497	2.119	4.818	0.988	66360
OLSR	65536 bps	120	13.556	2.000	3.059	0.999	52381
AntHocNet	65536 bps	5	17.025	3.798	6.611	0.139	6263

R	DR	NS	D	HC	J	PLR	RO
AntHocNet	65536 bps	10	20.675	3.518	8.658	0.178	6772
AntHocNet	65536 bps	20	27.928	3.801	10.650	0.213	8978
AntHocNet	65536 bps	30	27.510	3.962	11.005	0.275	10400
AntHocNet	65536 bps	50	32.066	3.846	12.550	0.321	14028
AntHocNet	65536 bps	70	33.975	3.841	13.911	0.380	17527
AntHocNet	65536 bps	90	29.555	4.122	12.662	0.429	18919
AntHocNet	65536 bps	120	21.048	4.174	9.875	0.460	19547

D.2.1.1 Delay Histogram Data

Table 10: Delay Histogram Data For EGM Mobility Model

End-to-End Delay (s)	# of Occurence	Node Speed (m/s)
0.001	85	90
0.002	141	90
0.003	151	90
0.004	169	90
0.005	197	90
0.006	168	90
0.007	135	90
0.008	131	90
0.009	154	90
0.01	105	90
0.011	100	90
0.012	67	90
0.013	46	90
0.014	56	90
0.015	49	90
0.016	30	90

End-to-End Delay (s)	# of Occurence	Node Speed (m/s)
0.017	20	90
0.018	17	90
0.019	29	90
0.02	26	90
0.021	25	90
0.022	19	90
0.023	15	90
0.024	13	90
0.025	15	90
0.026	12	90
0.027	19	90
0.028	8	90
0.029	12	90
0.03	14	90
0.031	9	90
0.032	13	90
0.033	11	90
0.034	10	90
0.035	6	90
0.036	3	90
0.037	5	90
0.038	12	90
0.039	11	90
0.04	10	90
0.041	6	90
0.042	7	90
0.043	8	90
0.044	6	90
0.045	5	90

End-to-End Delay (s)	# of Occurence	Node Speed (m/s)
0.046	5	90
0.047	3	90
0.048	4	90
0.049	3	90
0.05	7	90
0.051	6	90
0.052	7	90
0.053	10	90
0.054	5	90
0.055	4	90
0.056	4	90
0.057	6	90
0.058	2	90
0.059	7	90
0.06	4	90
0.061	3	90
0.062	1	90
0.064	2	90
0.065	3	90
0.066	2	90
0.067	3	90
0.068	3	90
0.069	3	90
0.071	1	90
0.072	1	90
0.073	4	90
0.074	2	90
0.076	1	90
0.077	1	90

End-to-End Delay (s)	# of Occurence	Node Speed (m/s)
0.078	3	90
0.079	1	90
0.08	4	90
0.081	1	90
0.082	1	90
0.086	2	90
0.087	1	90
0.088	2	90
0.089	1	90
0.091	1	90
0.092	1	90
0.093	2	90
0.096	1	90
0.1	1	90
0.101	2	90
0.102	1	90
0.103	1	90
0.105	1	90
0.107	1	90
0.109	1	90
0.111	1	90
0.112	1	90
0.117	2	90
0.119	3	90
0.12	1	90
0.121	2	90
0.135	1	90
0.142	1	90
0.149	2	90

End-to-End Delay (s)	# of Occurence	Node Speed (m/s)
0.159	2	90
0.172	1	90
0.173	1	90
0.176	1	90
0.19	1	90
0.199	2	90
0.239	1	90
0.256	1	90
0.261	1	90
0.28	1	90
0.283	1	90
0.287	1	90
0.294	1	90
0.311	1	90
0.317	1	90
0.326	1	90
0.327	1	90
0.331	1	90
0.368	1	90
0.385	1	90
0.39	1	90
0.405	1	90
0.416	1	90
0.437	1	90
0.463	1	90
0.515	1	90
0.518	1	90
0.519	1	90
0.525	1	90

End-to-End Delay (s)	# of Occurence	Node Speed (m/s)
0.53	1	90
0.534	2	90
0.536	1	90
0.542	1	90
0.544	1	90
0.545	1	90
0.547	1	90
0.555	1	90
0.56	1	90
0.563	1	90
0.569	1	90
0.581	1	90
0.585	1	90
0.59	1	90
0.591	1	90
0.592	1	90
0.603	1	90
0.61	1	90
0.613	1	90
0.614	1	90
0.615	1	90
0.643	1	90
0.658	1	90
0.667	1	90
0.671	2	90
0.678	1	90
0.679	1	90
0.68	1	90
0.689	1	90

End-to-End Delay (s)	# of Occurence	Node Speed (m/s)
0.698	1	90
0.7	1	90
0.724	1	90
0.733	1	90
0.749	1	90
0.754	1	90
0.756	1	90
0.775	1	90
0.821	1	90
0.83	1	90
0.853	1	90
0.877	1	90
0.904	1	90
0.942	1	90
1.013	1	90
1.015	1	90
1.018	1	90
1.02	1	90
1.021	1	90
1.026	1	90
1.027	2	90
1.032	2	90
1.034	2	90
1.047	1	90
1.057	1	90
1.065	1	90
1.099	1	90
1.109	1	90
1.113	1	90

End-to-End Delay (s)	# of Occurence	Node Speed (m/s)
1.146	1	90
1.172	1	90
1.184	1	90
1.195	1	90
1.222	1	90
1.225	1	90
1.237	1	90
1.239	1	90
1.249	1	90
1.3	1	90
1.438	1	90
1.46	1	90
1.515	1	90
1.522	1	90
1.541	1	90
1.611	1	90
1.681	1	90
1.694	1	90
1.712	1	90
1.723	1	90
1.732	1	90
1.748	1	90
1.793	1	90
1.935	1	90
1.952	1	90
2.005	1	90
2.02	1	90
2.023	1	90
2.109	1	90

End-to-End Delay (s)	# of Occurence	Node Speed (m/s)
2.191	1	90
2.207	1	90
2.221	1	90
2.247	1	90
2.284	1	90
2.432	1	90
2.502	1	90
2.517	1	90
2.527	1	90
2.607	1	90
2.623	1	90
2.684	1	90
2.687	2	90
2.719	1	90
2.742	1	90
2.747	1	90
2.929	1	90
2.982	1	90
2.997	1	90
3.016	1	90
3.104	1	90
3.113	1	90
3.128	1	90
3.179	1	90
3.182	1	90
3.183	1	90
3.217	1	90
3.235	1	90
3.246	1	90

End-to-End Delay (s)	# of Occurence	Node Speed (m/s)
3.426	1	90
3.469	1	90
3.48	1	90
3.505	1	90
3.509	1	90
3.572	1	90
3.602	2	90
3.62	1	90
3.673	1	90
3.674	1	90
3.681	1	90
3.715	1	90
3.729	1	90
3.745	1	90
3.748	1	90
3.922	1	90
3.974	1	90
3.998	1	90
4.058	1	90
4.098	1	90
4.118	1	90
4.171	1	90
4.213	1	90
4.244	2	90
4.551	1	90
4.655	1	90
4.711	1	90
4.728	1	90
4.744	1	90

End-to-End Delay (s)	# of Occurence	Node Speed (m/s)
5.209	1	90
5.243	1	90
5.707	1	90
5.742	1	90
6.149	1	90
6.205	1	90
6.617	1	90
6.702	1	90
7.11	1	90
7.198	1	90
7.604	1	90
7.716	1	90
8.092	1	90
8.211	1	90
8.247	1	90
8.559	1	90
8.736	1	90
9.054	1	90
9.224	1	90
0	32	120
0.001	60	120
0.002	113	120
0.003	53	120
0.004	49	120
0.005	44	120
0.006	61	120
0.007	69	120
0.008	48	120
0.009	61	120

End-to-End Delay (s)	# of Occurence	Node Speed (m/s)
0.01	31	120
0.011	32	120
0.012	39	120
0.013	15	120
0.014	17	120
0.015	17	120
0.016	14	120
0.017	12	120
0.018	12	120
0.019	11	120
0.02	6	120
0.021	4	120
0.022	4	120
0.023	6	120
0.024	12	120
0.025	2	120
0.026	5	120
0.027	6	120
0.028	5	120
0.03	8	120
0.031	3	120
0.032	2	120
0.033	2	120
0.034	5	120
0.035	3	120
0.036	2	120
0.037	5	120
0.038	5	120
0.039	3	120

End-to-End Delay (s)	# of Occurence	Node Speed (m/s)
0.04	6	120
0.041	1	120
0.042	1	120
0.044	3	120
0.045	4	120
0.046	2	120
0.048	1	120
0.049	5	120
0.05	1	120
0.051	2	120
0.052	3	120
0.053	3	120
0.054	4	120
0.055	1	120
0.057	1	120
0.058	1	120
0.059	2	120
0.06	6	120
0.062	2	120
0.063	1	120
0.064	1	120
0.067	1	120
0.068	2	120
0.069	5	120
0.071	1	120
0.074	1	120
0.076	2	120
0.08	1	120
0.083	1	120

End-to-End Delay (s)	# of Occurence	Node Speed (m/s)
0.084	2	120
0.085	2	120
0.091	1	120
0.092	1	120
0.094	1	120
0.098	1	120
0.112	1	120
0.117	1	120
0.124	1	120
0.145	1	120
0.175	1	120
0.18	1	120
0.225	1	120
0.259	1	120
0.265	1	120
0.28	1	120
0.313	1	120
0.314	1	120
0.483	1	120
0.526	1	120
0.542	1	120
0.548	1	120
0.553	1	120
0.573	1	120
0.574	1	120
0.58	1	120
0.581	1	120
0.642	1	120
0.679	1	120

End-to-End Delay (s)	# of Occurence	Node Speed (m/s)
0.721	1	120
0.764	1	120
0.774	1	120
0.891	1	120
0.979	1	120
1.021	1	120
1.038	1	120
1.041	1	120
1.042	1	120
1.125	1	120
1.153	1	120
1.199	1	120
1.261	1	120
1.475	1	120
1.57	1	120
1.609	1	120
1.695	1	120
1.757	1	120
1.97	1	120
2.006	1	120
2.465	1	120
2.584	1	120
2.606	1	120
2.83	1	120
2.956	1	120
3.037	1	120
3.074	1	120
3.093	1	120
3.323	1	120

End-to-End Delay (s)	# of Occurence	Node Speed (m/s)
3.451	1	120
3.53	1	120
3.564	1	120
3.587	1	120
3.815	1	120
4.018	1	120

D.2.2 Simulation Data for PPRZM Mobility Model

Table 11: Scenario2 Data for PPRZM Mobility Model

R	DR	NS	D	HC	J	PLR	RO
AODV	4096 bps	5	11.078	2.647	11.676	0.036	4109
AODV	4096 bps	10	15.641	2.529	18.642	0.055	6785
AODV	4096 bps	20	21.188	2.758	22.598	0.081	5970
AODV	4096 bps	30	26.728	2.709	26.415	0.112	6044
AODV	4096 bps	50	40.560	2.841	37.581	0.128	7631
AODV	4096 bps	70	31.704	2.694	32.464	0.181	8553
AODV	4096 bps	90	67.101	2.792	46.494	0.196	8873
AODV	4096 bps	120	78.605	2.646	52.679	0.216	10103
OLSR	4096 bps	5	6.617	2.371	4.818	0.063	171108
OLSR	4096 bps	10	8.058	2.277	6.453	0.162	177440
OLSR	4096 bps	20	15.811	2.353	15.730	0.371	161998
OLSR	4096 bps	30	17.584	2.300	16.894	0.491	153758
OLSR	4096 bps	50	38.432	2.335	34.258	0.650	131518
OLSR	4096 bps	70	45.347	2.313	48.159	0.742	119946
OLSR	4096 bps	90	63.132	2.177	58.858	0.768	100425
OLSR	4096 bps	120	79.246	2.041	73.636	0.825	86492
AntHocNet	4096 bps	5	2.792	3.352	1.235	0.010	4188

R	DR	NS	D	HC	J	PLR	RO
AntHocNet	4096 bps	10	5.346	3.402	3.320	0.045	4406
AntHocNet	4096 bps	20	8.777	3.302	7.569	0.063	5270
AntHocNet	4096 bps	30	11.188	3.322	10.442	0.082	5240
AntHocNet	4096 bps	50	16.293	3.430	17.613	0.112	5705
AntHocNet	4096 bps	70	20.112	3.465	24.166	0.141	5624
AntHocNet	4096 bps	90	26.955	3.267	35.452	0.166	5840
AntHocNet	4096 bps	120	27.874	3.178	33.377	0.195	6056
AODV	16384 bps	5	6.779	2.634	5.569	0.036	4341
AODV	16384 bps	10	9.463	2.531	7.563	0.069	5822
AODV	16384 bps	20	11.786	2.706	7.512	0.071	5305
AODV	16384 bps	30	12.514	2.747	9.279	0.130	7405
AODV	16384 bps	50	21.668	2.835	13.097	0.198	9363
AODV	16384 bps	70	29.557	2.661	17.013	0.235	10295
AODV	16384 bps	90	25.254	2.623	17.950	0.287	9889
AODV	16384 bps	120	54.048	2.566	23.782	0.301	11380
OLSR	16384 bps	5	4.489	2.387	3.268	0.051	170182
OLSR	16384 bps	10	8.269	2.306	5.609	0.182	175788
OLSR	16384 bps	20	10.400	2.381	6.859	0.372	156636
OLSR	16384 bps	30	15.267	2.353	10.918	0.479	144257
OLSR	16384 bps	50	21.405	2.296	16.130	0.662	126131
OLSR	16384 bps	70	29.193	2.243	22.433	0.749	107076
OLSR	16384 bps	90	30.895	2.102	24.448	0.788	94672
OLSR	16384 bps	120	42.384	2.095	32.087	0.846	81410
AntHocNet	16384 bps	5	2.348	3.399	1.066	0.013	4588
AntHocNet	16384 bps	10	5.508	3.310	3.358	0.051	4896
AntHocNet	16384 bps	20	6.666	3.515	4.568	0.087	5697
AntHocNet	16384 bps	30	7.202	3.476	4.510	0.116	5890
AntHocNet	16384 bps	50	10.827	3.635	7.311	0.128	7084
AntHocNet	16384 bps	70	10.671	3.608	7.043	0.149	7572

R	DR	NS	D	HC	J	PLR	RO
AntHocNet	16384 bps	90	12.354	3.518	8.902	0.169	8142
AntHocNet	16384 bps	120	14.735	3.443	10.502	0.199	8650
AODV	65536 bps	5	64.495	2.674	15.806	0.161	12853
AODV	65536 bps	10	37.620	2.362	12.862	0.229	14694
AODV	65536 bps	20	23.031	2.676	7.949	0.307	8775
AODV	65536 bps	30	25.152	2.689	8.630	0.330	10303
AODV	65536 bps	50	30.717	2.699	9.192	0.371	10376
AODV	65536 bps	70	32.196	2.508	8.862	0.462	10988
AODV	65536 bps	90	34.999	2.398	10.284	0.460	11141
AODV	65536 bps	120	37.400	2.330	11.393	0.497	12475
OLSR	65536 bps	5	20.550	2.449	8.291	0.106	161360
OLSR	65536 bps	10	15.143	2.301	7.819	0.219	163973
OLSR	65536 bps	20	18.820	2.430	10.523	0.420	137737
OLSR	65536 bps	30	20.015	2.392	10.903	0.540	125433
OLSR	65536 bps	50	24.385	2.334	13.256	0.708	104725
OLSR	65536 bps	70	28.923	2.289	13.834	0.772	89792
OLSR	65536 bps	90	36.585	2.148	16.314	0.811	81035
OLSR	65536 bps	120	47.943	1.931	19.460	0.851	70141
AntHocNet	65536 bps	5	25.587	3.071	9.177	0.115	5988
AntHocNet	65536 bps	10	22.721	3.264	10.237	0.154	6711
AntHocNet	65536 bps	20	27.456	3.505	10.063	0.221	8632
AntHocNet	65536 bps	30	30.252	3.477	12.511	0.248	9930
AntHocNet	65536 bps	50	30.187	3.380	11.747	0.291	11268
AntHocNet	65536 bps	70	27.087	3.449	10.847	0.315	12211
AntHocNet	65536 bps	90	30.537	3.490	12.278	0.355	12869
AntHocNet	65536 bps	120	25.649	3.268	12.005	0.405	14574

D.2.3 Simulation Data for RWP Mobility Model

Table 12: Scenario2 Data for RWP Mobility Model

R	DR	NS	D	HC	J	PLR	RO
AODV	4096 bps	5	9.378	2.570	11.112	0.045	4214
AODV	4096 bps	10	22.073	2.459	25.626	0.068	6944
AODV	4096 bps	20	21.388	2.538	24.765	0.102	10914
AODV	4096 bps	30	26.673	2.715	29.011	0.085	10079
AODV	4096 bps	50	44.220	2.687	50.620	0.131	13189
AODV	4096 bps	70	43.368	2.585	45.530	0.151	15506
AODV	4096 bps	90	68.521	2.492	66.253	0.175	17078
AODV	4096 bps	120	61.246	2.330	51.988	0.229	16724
OLSR	4096 bps	5	7.370	2.307	6.271	0.050	171889
OLSR	4096 bps	10	11.760	2.253	10.242	0.144	185618
OLSR	4096 bps	20	14.431	2.257	13.248	0.328	187884
OLSR	4096 bps	30	16.267	2.330	15.584	0.446	178659
OLSR	4096 bps	50	32.192	2.275	25.041	0.635	160232
OLSR	4096 bps	70	26.135	2.012	26.537	0.720	138132
OLSR	4096 bps	90	36.986	1.850	26.101	0.843	115551
OLSR	4096 bps	120	71.580	1.917	62.382	0.881	86566
AntHocNet	4096 bps	5	3.084	3.309	1.602	0.008	4175
AntHocNet	4096 bps	10	5.876	3.167	4.892	0.059	4878
AntHocNet	4096 bps	20	9.877	3.386	8.451	0.065	4991
AntHocNet	4096 bps	30	12.701	3.244	12.915	0.091	5070
AntHocNet	4096 bps	50	21.277	3.224	26.652	0.115	5674
AntHocNet	4096 bps	70	26.440	3.066	35.920	0.165	6029
AntHocNet	4096 bps	90	30.175	2.985	42.987	0.188	6043
AntHocNet	4096 bps	120	32.907	2.888	45.237	0.232	5895
AODV	16384 bps	5	8.532	2.629	6.266	0.044	4653
AODV	16384 bps	10	17.113	2.436	13.909	0.073	8298
AODV	16384 bps	20	11.286	2.508	9.602	0.086	11669

R	DR	NS	D	HC	J	PLR	RO
AODV	16384 bps	30	17.089	2.543	14.809	0.151	11517
AODV	16384 bps	50	22.591	2.511	20.939	0.180	14392
AODV	16384 bps	70	21.202	2.373	19.079	0.185	17078
AODV	16384 bps	90	53.007	2.514	27.920	0.225	19927
AODV	16384 bps	120	64.972	2.294	24.084	0.276	18528
OLSR	16384 bps	5	5.278	2.298	4.008	0.054	170153
OLSR	16384 bps	10	8.372	2.228	5.961	0.177	181217
OLSR	16384 bps	20	11.400	2.277	7.729	0.332	180614
OLSR	16384 bps	30	16.663	2.335	13.518	0.474	171611
OLSR	16384 bps	50	22.406	2.258	20.069	0.639	154387
OLSR	16384 bps	70	15.264	2.023	12.859	0.737	128387
OLSR	16384 bps	90	21.930	1.854	17.187	0.834	107581
OLSR	16384 bps	120	24.590	1.808	15.457	0.905	81564
AntHocNet	16384 bps	5	2.071	3.125	1.302	0.013	4236
AntHocNet	16384 bps	10	9.361	3.195	6.001	0.072	5103
AntHocNet	16384 bps	20	6.655	3.374	4.975	0.081	5475
AntHocNet	16384 bps	30	8.961	3.407	6.286	0.111	5703
AntHocNet	16384 bps	50	14.454	3.371	10.808	0.141	7064
AntHocNet	16384 bps	70	12.684	3.075	9.564	0.170	7524
AntHocNet	16384 bps	90	15.507	3.067	11.781	0.196	8030
AntHocNet	16384 bps	120	15.570	3.070	11.771	0.226	8807
AODV	65536 bps	5	71.251	2.598	19.562	0.173	12835
AODV	65536 bps	10	56.454	2.457	15.397	0.186	15235
AODV	65536 bps	20	48.623	2.484	15.799	0.219	15007
AODV	65536 bps	30	42.776	2.570	13.739	0.347	13888
AODV	65536 bps	50	45.536	2.415	13.847	0.389	14462
AODV	65536 bps	70	48.974	2.269	15.331	0.377	17874
AODV	65536 bps	90	45.530	2.231	14.369	0.450	17497
AODV	65536 bps	120	58.608	2.117	15.817	0.476	19026

R	DR	NS	D	HC	J	PLR	RO
OLSR	65536 bps	5	16.290	2.313	5.992	0.095	164043
OLSR	65536 bps	10	14.951	2.263	7.866	0.188	170068
OLSR	65536 bps	20	23.542	2.200	13.359	0.404	161562
OLSR	65536 bps	30	21.139	2.313	11.853	0.485	150142
OLSR	65536 bps	50	26.437	2.342	15.906	0.725	123685
OLSR	65536 bps	70	22.352	2.024	12.518	0.781	106265
OLSR	65536 bps	90	23.257	1.889	12.736	0.871	87922
OLSR	65536 bps	120	43.044	1.908	21.270	0.920	66745
AntHocNet	65536 bps	5	19.144	3.153	8.235	0.079	5332
AntHocNet	65536 bps	10	49.705	3.024	15.084	0.163	6544
AntHocNet	65536 bps	20	33.324	3.123	12.344	0.219	8457
AntHocNet	65536 bps	30	51.564	3.298	19.905	0.267	9469
AntHocNet	65536 bps	50	32.158	3.163	12.772	0.296	10824
AntHocNet	65536 bps	70	27.192	3.116	11.619	0.355	11495
AntHocNet	65536 bps	90	33.553	2.975	13.770	0.398	12668
AntHocNet	65536 bps	120	35.093	2.927	14.732	0.412	13884